Solaris のシステム管理 (セキュリティサービス)

Part No: 819–0383–19
2011 年 8 月

ORACLE

120418@25097

目次

はじめに

『Solaris のシステム管理 (セキュリティサービス)』は、Oracle Solaris オペレーティングシステム (Oracle Solaris) の管理マニュアルの一部です。このドキュメントは、現在のリリースをすでにインストールしていて、ネットワークソフトウェアの設定を終了していることを前提としています。Oracle Solaris オペレーティングシステムは、Oracle Solaris Secure Shell など、多くの機能を含む Oracle Solaris 製品ファミリの一部です。

注 – この Oracle Solaris のリリースでは、SPARC および x86 系列のプロセッサアーキテクチャーを使用するシステムをサポートしています。サポートされるシステムは、Oracle Solaris OS: Hardware Compatibility Lists に記載されています。本書では、プラットフォームにより実装が異なる場合は、それを特記します。

本書の x86 に関連する用語については、次を参照してください。

- x86 は、64 ビットおよび 32 ビットの x86 互換製品系列を指します。
- x64 は特に 64 ビット x86 互換 CPU を指します。
- 32 ビット x86 は、x86 ベースのシステムに関する 32 ビット特有の情報を指します。

サポートされるシステムについては、Oracle Solaris OS: Hardware Compatibility Lists を参照してください。

対象読者

このドキュメントは、Oracle Solaris のシステム管理者を対象にしています。このドキュメントを利用するにあたっては、UNIX のシステム管理について 3 年以上の経験が必要です。UNIX システム管理のトレーニングコースに参加することをお勧めします。

Solaris システム管理マニュアルセットの構成

システム管理マニュアルセットに含まれる各マニュアルとその内容は、次のとおりです。

マニュアルのタイトル	トピック
『Solaris のシステム管理 (基本編)』	ユーザーアカウントとグループ、サーバーとクライアントのサポート、システムのシャットダウンとブート、およびサービスの管理
『Solaris のシステム管理 (上級編)』	端末とモデムの設定、システムリソースの管理 (ディスク割り当て、アカウンティング、および crontab ファイルの管理)、 システムプロセスの管理、および Oracle Solaris ソフトウェアの障害追跡
『Solaris のシステム管理 (デバイスとファイルシステム)』	リムーバブルメディア、ディスクとデバイス、ファイルシステム、およびデータのバックアップと復元
『Solaris のシステム管理 (IP サービス)』	TCP/IP ネットワークの管理、IPv4 および IPv6 アドレスの管理、DHCP、 IP セキュリティー、IKE、 IP フィルタ、モバイル IP、 IP ネットワークのマルチパス化 (IPMP)、 および IPQoS
『Solaris のシステム管理 (ネーミングとディレクトリサービス : DNS、NIS、LDAP 編)』	DNS、 NIS、 および LDAP のネーミングとディレクトリサービス (NIS から LDAP への移行、および NIS+ から LDAP への移行を含む)
『Solaris のシステム管理 (ネーミングとディレクトリサービス : NIS+ 編)』	NIS+ のネーミングとディレクトリサービス
『Solaris のシステム管理 (ネットワークサービス)』	Web キャッシュサーバー、時間関連サービス、ネットワークファイルシステム (NFS と autofs)、 メール、SLP、および PPP
『Solaris のシステム管理 (印刷)』	印刷に関するトピックや、サービス、ツール、プロトコル、およびテクノロジを使って印刷サービスおよびプリンタを設定および管理する方法
『Solaris のシステム管理 (セキュリティサービス)』	監査、デバイス管理、ファイルセキュリティー、BART、Kerberos サービス、PAM、 Oracle Solaris 暗号化フレームワーク、特権、RBAC、 SASL、 および Oracle Solaris Secure Shell
『Oracle Solaris のシステム管理 (Oracle Solaris コンテナ : 資源管理と Oracle Solaris ゾーン)』	リソース管理に関連する計画と作業、拡張アカウンティング、リソース制御、フェアシェアスケジューラ (FSS)、 資源上限デーモン (rcapd) による物理メモリーの制御、および資源プール (Solaris Zones ソフトウェア区分技術と lx ブランドゾーンによる仮想化)

マニュアルのタイトル	トピック
『Oracle Solaris ZFS 管理ガイド』	ZFS ストレージプールおよびファイルシステムの作成と管理、スナップショット、クローン、バックアップ、アクセス制御リスト (ACL) による ZFS ファイルの保護、ゾーンがインストールされた Oracle Solaris システム上での ZFS の使用、エミュレートされたボリューム、およびトラブルシューティングとデータ回復
『Oracle Solaris Trusted Extensions 管理の手順』	Oracle Solaris Trusted Extensions 機能固有のシステム管理
『Oracle Solaris Trusted Extensions 構成ガイド』	Solaris 10 5/08 リリース以降での、Oracle Solaris Trusted Extensions 機能の計画、有効化、および初期設定の方法

関連する Sun 以外の Web サイト情報

このドキュメントでは、Sun 以外の URL を挙げ、関連する補足情報を示す場合があります。

このソフトウェアまたはハードウェア、そしてドキュメントは、第三者のコンテンツ、製品、サービスへのアクセス、あるいはそれらに関する情報を提供することがあります。オラクル社およびその関連会社は、第三者のコンテンツ、製品、サービスに関して一切の責任を負わず、いかなる保証もいたしません。オラクル社およびその関連会社は、第三者のコンテンツ、製品、サービスへのアクセスまたは使用によって損失、費用、あるいは損害が発生しても一切の責任を負いかねます。

Oracle サポートへのアクセス

Oracle のお客様は、My Oracle Support を通じて電子的なサポートを利用することができます。詳細は、`http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` を参照してください。聴覚に障害をお持ちの場合は、`http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` を参照してください。

表記上の規則

このマニュアルでは、次のような字体や記号を特別な意味を持つものとして使用します。

表 P–1 表記上の規則

字体または記号	意味	例
AaBbCc123	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例を示します。	.login ファイルを編集します。 ls -a を使用してすべてのファイルを表示します。 system%
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	system% **su** password:
AaBbCc123	変数を示します。実際に使用する特定の名前または値で置き換えます。	ファイルを削除するには、rm *filename* と入力します。
『 』	参照する書名を示します。	『コードマネージャ・ユーザーズガイド』を参照してください。
「 」	参照する章、節、ボタンやメニュー名、強調する単語を示します。	第 5 章「衝突の回避」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合に、継続を示します。	sun% **grep '^#define ** **XV_VERSION_STRING'**

Oracle Solaris OS に含まれるシェルで使用する、UNIX のデフォルトのシステムプロンプトとスーパーユーザープロンプトを次に示します。コマンド例に示されるデフォルトのシステムプロンプトは、Oracle Solaris のリリースによって異なります。

- C シェル

  machine_name% **command y|n** [*filename*]

- C シェルのスーパーユーザー

  machine_name# **command y|n** [*filename*]

- Bash シェル、Korn シェル、および Bourne シェル

  $ **command y|n** [*filename*]

- Bash シェル、Korn シェル、および Bourne シェルのスーパーユーザー

  # **command y|n** [*filename*]

[ ] は省略可能な項目を示します。上記の例は、*filename* は省略してもよいことを示しています。

| は区切り文字 (セパレータ) です。この文字で分割されている引数のうち 1 つだけを指定します。

キーボードのキー名は英文で、頭文字を大文字で示します (例: Shift キーを押します)。ただし、キーボードによっては Enter キーが Return キーの動作をします。

ダッシュ (-) は 2 つのキーを同時に押すことを示します。たとえば、Ctrl-D は Control キーを押したまま D キーを押すことを意味します。

一般規則

- このマニュアルでは、英語環境での画面イメージを使っています。このため、実際に日本語環境で表示される画面イメージとこのマニュアルで使っている画面イメージが異なる場合があります。本文中で画面イメージを説明する場合には、日本語のメニュー、ボタン名などの項目名と英語の項目名が、適宜併記されています。

パート I

セキュリティーの概要

このドキュメントでは、Oracle Solaris オペレーティングシステム のセキュリティー支援機能について説明します。セキュリティー機能のシステム管理者とユーザーを対象としています。第 1 章「セキュリティーサービス (概要)」では、このドキュメントのトピックについて説明します。

◆ ◆ ◆ 第 1 章

1 セキュリティーサービス (概要)

Oracle Solaris オペレーティングシステム (Oracle Solaris OS) のセキュリティーを保持するため、ソフトウェアでは次のような機能を提供しています。

- 34 ページの「システムセキュリティー」 – 侵入を防止する、誤用されないようにマシンリソースやデバイスを保護する、ユーザーや侵入者によって悪意のある変更または意図的でない変更が行われないようにファイルを保護する

  ローカルシステムセキュリティーについては、第 2 章「マシンセキュリティーの管理 (概要)」を参照してください。
- 35 ページの「暗号化サービス」 - 送信者と、宛先として指定された受信者だけが内容を読むことができるようにデータにスクランブルをかけるとともに、暗号化プロバイダおよび公開鍵オブジェクトを管理する
- 36 ページの「認証サービス」 – 安全にユーザーを識別する。ユーザー名とその証明書 (通常はパスワード) を要求する
- 37 ページの「暗号化による認証」 – 認証されたユーザーまたはグループが通信するときに、傍受、改ざん、または偽装を防ぐ
- 37 ページの「監査」 - ファイルアクセス、セキュリティー関連のシステムコール、および認証の失敗など、セキュリティーに変化が起きた場所を識別してシステムに通知する
- 37 ページの「セキュリティーポリシー」 - 単体のシステムまたはシステムネットワークを対象としたセキュリティーガイドラインの設計と実装

システムセキュリティー

システムセキュリティーとは、システムのリソースが適切に使用されるようにするための手段です。アクセス制御を行うと、システム上のリソースにアクセスできるユーザーを制限することができます。システムセキュリティーとアクセス制御を目的とした Oracle Solaris OS の機能には次のようなものがあります。

- ログイン管理ツール – ユーザーのログイン能力の監視と制御を行うコマンド。66 ページの「ログインとパスワードの保護 (作業マップ)」を参照してください。
- ハードウェアアクセス – PROM へのアクセスを制限するコマンドや、システムを起動できるユーザーを制限するコマンド。81 ページの「SPARC: システムハードウェアに対するアクセスの制御 (作業マップ)」を参照してください。
- リソースアクセス – マシンリソースが適切にかつ最大限に利用されるように対策を施すとともに、それらのリソースの誤用を最小限に抑えるツールと戦略。52 ページの「マシンリソースへのアクセス制御」を参照してください。
- 役割によるアクセス制御 **(RBAC)** – 特殊な制限付きユーザーアカウントを作成するためのアーキテクチャー。特定の管理作業の実行を許可します。188 ページの「役割によるアクセス制御 (概要)」を参照してください。
- 特権 – 処理を実行するプロセスにおける個々の権限。これらのプロセス権限はカーネル内で適用されます。199 ページの「特権 (概要)」を参照してください。
- デバイス管理 – UNIX の権限ですでに行われているデバイス保護をさらに強化するデバイス「ポリシー」。デバイス「割り当て」は、マイクや CD-ROM ドライブなどの周辺機器に対するアクセスを制御します。割り当てを解除する際には、デバイスクリーンスクリプトによってデバイスから任意のデータを削除できます。49 ページの「デバイスアクセスの制御」を参照してください。
- 基本監査報告機能 **(BART)** – システム上のファイルの属性を示す、「目録」と呼ばれるスナップショット。一定期間にわたって複数のシステムまたは単一のシステム上のいくつかの目録を比較することで、ファイルの変化を監視し、セキュリティーリスクを抑えることができます。第 5 章「基本監査報告機能の使用方法 (作業)」を参照してください。
- ファイルアクセス権 – ファイルまたはディレクトリの属性。アクセス権を使用すれば、ファイルの読み取り、書き込み、実行、あるいはディレクトリの検索を行えるユーザーおよびグループを制限できます。第 6 章「ファイルアクセスの制御 (作業)」を参照してください。
- セキュリティー強化スクリプト – スクリプトを使用することにより、多数のシステムファイルとパラメータを調整し、セキュリティーの危険性を減少させます。第 7 章「自動セキュリティー拡張ツールの使用 (手順)」を参照してください。

暗号化サービス

暗号化はデータの符号化と復号化の技術であり、整合性、機密性、および信頼性を維持するために使用されます。整合性とは、データが改ざんされていないことを意味します。機密性は、データがほかのユーザーによって読み取られることがないことを意味します。データの信頼性は、送信された内容と受信した内容が同じであることを意味します。ユーザーの認証は、そのユーザーが自己の証明となる 1 つ以上の情報を提供したことを意味します。認証メカニズムは、データソースやユーザー識別情報などを数学的に検証します。暗号化メカニズムは、偶然にデータを目にした人がそのデータを読み取ることができないようにデータにスクランブルをかけます。暗号化サービスは、アプリケーションやユーザーに認証と暗号化のメカニズムを提供します。

暗号化アルゴリズムは、ハッシュ、チェーンなどの数学技法を使って解読が極めて困難な暗号を作成します。認証メカニズムは、送信者と受信者がデータから同一の数値を算出することを要求します。暗号化メカニズムは、送信者と受信者が暗号化方式に関する情報を共有することに基づいています。この情報によって、送信者と受信者だけがメッセージを復号化できます。Oracle Solaris は、一元化された暗号化フレームワークと、特定のアプリケーション専用の暗号化メカニズムを提供します。

- **Oracle Solaris** 暗号化フレームワーク – RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki) の標準に基づく、カーネルレベルとユーザーレベルのコンシューマ向けの暗号化サービスの集中型フレームワーク。用途には、パスワード、IP セキュリティープロトコル、Sun 以外のアプリケーションなどがあります。このフレームワークでは、ハードウェアとソフトウェアのソースを暗号化用に一元化します。PKCS #11 ライブラリは、Sun 以外の開発者がアプリケーションの暗号化仕様を実装するための API を提供します。第 13 章「Oracle Solaris の暗号化フレームワーク (概要)」を参照してください。
- アプリケーションごとの暗号化メカニズム –
  - Secure RPC における DES の使用については、331 ページの「Secure RPC の概要」を参照してください。
  - Kerberos サービスにおける DES、 3DES、 AES、 および ARCFOUR の使用については、第 21 章「Kerberos サービスについて」を参照してください。
  - Secure Shell における RSA、 DSA、 および暗号 (Blowfish など) の使用については、第 19 章「Oracle Solaris Secure Shell の使用 (手順)」を参照してください。
  - パスワードにおける暗号化アルゴリズムの使用については、74 ページの「パスワードアルゴリズムの変更 (作業マップ)」を参照してください。

Solaris 10 8/07 リリースから、鍵管理フレームワーク (KMF) は、ポリシー、鍵、証明書などの公開鍵オブジェクトを集中管理するためのユーティリティーを提供しています。KMF は、これらのオブジェクトを OpenSSL、 NSS、 および PKCS #11 公開鍵技術向けに管理します。第 15 章「Oracle Solaris 鍵管理フレームワーク」を参照してください。

認証サービス

認証とは、定義済みの条件に基づいてユーザーまたはサービスを識別するメカニズムのことです。認証サービスには、単純な認証システム (名前とパスワードの組み合わせ) から複雑な暗号化認証システム (トークンカード、生体認証など) まで、さまざまな形態があります。強力な認証メカニズムは、ユーザーだけが知っている情報や検証可能な個人情報を使用します。ユーザー名は、ユーザーが知っている情報の一例です。検証可能な情報には、スマートカードや指紋などがあります。認証に関連した Oracle Solaris 機能には次のようなものがあります。

- **Secure RPC** - NFS マウントやネームサービス (NIS や NIS+ など) の保護に Diffie-Hellman プロトコル を使用する認証メカニズム。331 ページの「Secure RPC の概要」を参照してください。
- プラグイン可能認証モジュール **(PAM)** – システムに入るためのサービスをコンパイルし直すことなく、それらのサービスにさまざまな認証技術を加えることができるようにするフレームワーク。システムエントリサービスには、`login` や `ftp` などがあります。第 17 章「PAM の使用」を参照してください。
- 簡易認証セキュリティー層 **(SASL)** – ネットワークプロトコルに認証サービスとセキュリティーサービスを提供するフレームワーク。第 18 章「SASL の使用」を参照してください。
- **Secure Shell** – セキュリティー保護が行われていないネットワーク上で通信を暗号化する、セキュリティー保護付きの遠隔ログインと転送プロトコル。第 19 章「Oracle Solaris Secure Shell の使用 (手順)」を参照してください。
- **Kerberos** サービス – 認証による暗号化を行うクライアントサーバーアーキテクチャー。第 21 章「Kerberos サービスについて」を参照してください。
- **Solaris** スマートカード – マイクロプロセッサとメモリーが組み込まれたプラスチックのカード。システムにアクセスするときに、カードリーダーを使用します。『Solaris スマートカードの管理』を参照してください。

暗号化による認証

暗号化による認証は通信を安全に行う基本です。認証を利用して、送信元と送信先が正しいユーザーまたはグループであることを保証します。通信は、送信元で暗号化され、送信先で復号化されます。暗号化されていれば、侵入者が通信を傍受できたとしても、その内容が解読されることはありません。通信を安全に行うための Oracle Solaris の機能には次のようなものがあります。

- **Secure Shell** – データ転送と対話型ユーザーのネットワークセッションを、盗聴、セッションハイジャック、および「man-in-the-middle」攻撃から保護するプロトコルの 1 つ。公開鍵暗号化によって、強力な認証を提供します。X ウィンドウシステムなどのネットワークサービスは、Secure Shell 接続によって安全にトンネル化することで、セキュリティーが向上します。第 19 章「Oracle Solaris Secure Shell の使用 (手順)」を参照してください。
- **Kerberos** サービス – 暗号化による認証を行うクライアントサーバーアーキテクチャー。第 21 章「Kerberos サービスについて」を参照してください。
- インターネットプロトコルセキュリティーアーキテクチャー **(IPsec)** – IP データグラムを保護するアーキテクチャー。機密性、強力なデータ完全性、データ認証、部分的なシーケンス完全性を実現します。『Solaris のシステム管理 (IP サービス)』の第 19 章「IP セキュリティーアーキテクチャー (概要)」を参照してください。

監査

監査は、システムのセキュリティーと保全性に関する基本概念です。監査は、システムの動作とイベントの履歴を検査して、発生した処理を確認するプロセスです。この履歴は、処理内容、処理日時、処理を行った人物、処理による影響をログとして記録したものです。第 28 章「Oracle Solaris 監査 (概要)」を参照してください。

セキュリティーポリシー

セキュリティーポリシーまたはポリシーという表現は、このドキュメントでは組織のセキュリティーガイドラインと同じ意味で使用されています。実際のサイトのセキュリティーポリシーは、処理される情報の重要度や未承認アクセスから情報を保護する手段を定義する規則セットです。Secure Shell、 認証、RBAC、 承認、特権、リソース制御などのセキュリティー技術は、情報を保護する手段となります。

セキュリティー技術の中には、それらの個々の実装側面を表現する上でワードポリシーを使用するものもあります。たとえば、Oracle Solaris は監査ポリシーの一部の設定で監査ポリシーオプションを使用します。次の表は、個々の実装側面の表現にワードポリシーを使用する機能に関連した用語解説、マニュアルページ、および情報を示しています。

表 1–1　Oracle Solaris OS におけるポリシーの利用

用語定義	選択されるマニュアルページ	詳細情報
audit policy	audit_control(4), audit_user(4), auditconfig(1M)	第 28 章「Oracle Solaris 監査 (概要)」
policy in the cryptographic framework	cryptoadm(1M)	第 13 章「Oracle Solaris の暗号化フレームワーク (概要)」
device policy	getdevpolicy(1M)	49 ページの「デバイスアクセスの制御」
Kerberos policy	krb5.conf(4)	第 25 章「Kerberos 主体とポリシーの管理 (手順)」
network policies	ipfilter(5), ifconfig(1m), ike.config(4), ipsecconf(1M), routeadm(1M)	『Solaris のシステム管理 (IP サービス)』のパート IV「IP セキュリティー」
password policy	passwd(1), nsswitch.conf(4), crypt.conf(4), policy.conf(4)	43 ページの「ログイン制御の管理」
policy for public key technologies	kmfcfg(1)	第 15 章「Oracle Solaris 鍵管理フレームワーク」
RBAC policy	rbac(5). policy.conf(4)	256 ページの「policy.conf ファイル」

パート II

システム、ファイル、およびデバイスのセキュリティー

このパートでは、ネットワークに接続されていないシステムで設定されるセキュリティーについて説明します。各章では、ディスク、ファイル、および周辺機器へのアクセスの計画、監視、および制御について説明します。

◆ ◆ ◆ 第 2 章

マシンセキュリティーの管理 (概要)

マシンの情報を安全に保持することは、システム管理者の重要な責任です。この章では、マシンセキュリティー管理の概要を説明します。

この章の内容は次のとおりです。

Solaris 10 リリースにおけるマシンセキュリティーの向上

Solaris 9 のリリース以降、システムセキュリティーを向上させるために次の機能が追加されました。

- 強力なパスワード暗号化機能を利用できます。この機能はユーザーによる設定が可能です。詳細は、45 ページの「パスワードの暗号化」を参照してください。
- デバイスポリシーが特権によって強化されています。詳細は、50 ページの「デバイスポリシー (概要)」を参照してください。

  デバイス割り当てについては、今後の Oracle Solaris リリースで `/etc/security/dev` ディレクトリがサポートされなくなる可能性があります。
- 基本監査報告機能 (BART) は、システム上のファイルの信頼性を監視するために使用できます。詳細は、第 5 章「基本監査報告機能の使用方法 (作業)」を参照してください。
- 強力な暗号化でファイルを保護できるようになりました。詳細は、57 ページの「暗号化によるファイルの保護」を参照してください。

- 特権により、カーネルレベルでプロセスの権限を指定できます。詳細は、199 ページの「特権 (概要)」を参照してください。
- 暗号化フレームワークにより、プロバイダやコンシューマを対象にした暗号化サービスが一元化されます。詳細は、第 13 章「Oracle Solaris の暗号化フレームワーク (概要)」を参照してください。
- PAM フレームワークで、さまざまなプログラムに対応できる機能 (Secure Shell など) が提供されます。詳細は、345 ページの「Solaris 10 リリースにおける PAM への変更」を参照してください。
- マシンリソースに対して Oracle Solaris ゾーンの指定やリソース管理制御アクセスが行えます。詳細は、『Oracle Solaris のシステム管理 (Oracle Solaris コンテナ : 資源管理と Oracle Solaris ゾーン)』を参照してください。

コンピュータシステムへのアクセスを制御する

ワークスペースでは、サーバーに接続されたすべてのマシンを 1 つの大規模多重システムと見なすことができます。システム管理者は、この大規模なシステムのセキュリティー管理に責任があります。システム管理者は、ネットワークの外部からの侵入を防ぐ必要があります。また、ネットワーク内部のコンピュータ上のデータの完全性を確保する必要もあります。

ファイルレベルにおいて、Oracle Solaris には標準セキュリティー機能が組み込まれており、ファイル、ディレクトリ、およびデバイスを保護するために使用できます。システムレベルとネットワークレベルでは、セキュリティーの内容はほぼ同じです。セキュリティー防御の第一線は、システムへのアクセスを制御することです。

次の方法でシステムへのアクセスを制御または監視できます。

- 42 ページの「物理的なセキュリティーの管理」
- 43 ページの「ログイン制御の管理」
- 49 ページの「デバイスアクセスの制御」
- 52 ページの「マシンリソースへのアクセス制御」
- 57 ページの「ファイルアクセスの制御」
- 59 ページの「ネットワークアクセスの制御」
- 64 ページの「セキュリティー問題の報告」

物理的なセキュリティーの管理

システムへのアクセスを制御するには、コンピュータ環境の物理的なセキュリティーを管理する必要があります。たとえば、システムにログインしたままこれを放置することは未承認アクセスを招く原因になります。侵入者がオペレーティング

システムやネットワークにアクセスしないとも限らないからです。コンピュータの周辺環境やコンピュータハードウェアは、不当なアクセスから物理的に保護される必要があります。

システム管理者は、ハードウェア設定に対する不当なアクセスから SPARC システムを保護することができます。`eeprom` コマンドを使って、パスワードがないと PROM にアクセスできないようにしてください。詳細は、81 ページの「ハードウェアアクセスのパスワードを必須にする方法」を参照してください。

ログイン制御の管理

システムやネットワークへの無許可のログインも防止する必要があります。この制限は、パスワード割り当てとログイン制御によって行うことができます。システム上のすべてのアカウントには、パスワードが必要です。パスワードはシンプルな認証メカニズムです。アカウントにパスワードを設定しないと、ユーザー名を推測できる侵入者であれば誰でもネットワーク全体にアクセスできることになります。力ずくの野蛮な攻撃を許さないためには、強力なパスワードアルゴリズムが必要です。

ユーザーがシステムにログインすると、`login` コマンドは `/etc/nsswitch.conf` ファイル内の情報に従って、該当するネームサービスまたはディレクトリサービスのデータベースを照会します。このファイルには次のエントリを含めることができます。

- `files` – ローカルシステムの `/etc` ファイルを指定します
- `ldap` – LDAP サーバーの LDAP ディレクトリサービスを指定します
- `nis` – NIS マスターサーバーの NIS データベースを指定します
- `nisplus` – NIS+ root サーバーの NIS+ データベースを指定します

`nsswitch.conf` ファイルの詳細は、`nsswitch.conf(4)` のマニュアルページを参照してください。ネームサービスおよびディレクトリサービスについては、『Solaris のシステム管理 (ネーミングとディレクトリサービス : DNS、NIS、LDAP 編)』または『Solaris のシステム管理 (ネーミングとディレクトリサービス : NIS+ 編)』を参照してください。

`login` コマンドは、指定されたユーザー名とパスワードを検証します。ユーザー名がパスワードファイルにないと、`login` コマンドはシステムへのアクセスを拒否します。あるいは、指定されたユーザー名に対するパスワードが正しくないと、`login` コマンドはシステムへのアクセスを拒否します。有効なユーザー名とそれに対応するパスワードが入力されれば、システムはシステムへのアクセスをユーザーに認可します。

PAM モジュールには、システムへのログインが正常に完了したあとでアプリケーションへのログインをスムーズに行わせる効果があります。詳細は、第 17 章「PAM の使用」を参照してください。

Oracle Solaris システムには、精巧な認証メカニズムと承認メカニズムが備わっています。ネットワークレベルでの認証メカニズムや承認メカニズムについては、60 ページの「遠隔アクセスの認証と承認」を参照してください。

パスワード情報の管理

ユーザーはシステムにログインするときに、ユーザー名とパスワードの両方を入力する必要があります。ログイン名は公開されていますが、パスワードは秘密にしなければなりません。ユーザーは、自分のパスワードを他人に知られてはいけません。また、各自のパスワードを慎重に選択し、頻繁に変更するようにしなければなりません。

パスワードは、最初にユーザーアカウントを設定するときに作成されます。ユーザーアカウントのセキュリティーを管理するために、パスワード有効期限を設定し、パスワードを定期的に強制変更することができます。また、ユーザーアカウントを無効にして、パスワードをロックすることもできます。パスワードの管理の詳細は、『Solaris のシステム管理 (基本編)』の第 4 章「ユーザーアカウントとグループの管理 (概要)」と、`passwd(1)` のマニュアルページを参照してください。

ローカルパスワード

ネットワークでローカルファイルを使用してユーザーの認証を行なっている場合、パスワード情報はシステムの `/etc/passwd` ファイルと `/etc/shadow` ファイルに保持されます。ユーザー名などの情報は、`/etc/passwd` ファイルに保持されます。暗号化されたパスワードは、*/etc/shadow* という「シャドウ」ファイルに保持されます。このセキュリティー方式によって、暗号化されたパスワードにアクセスされることを防ぎます。`/etc/passwd` ファイルは、システムにログインするユーザーであれば誰でも使用できますが、`/etc/shadow` ファイルを読み取ることができるのはスーパーユーザー、またはスーパーユーザーと同等の役割だけです。`passwd` コマンドを使用すると、ローカルシステム上のユーザーのパスワードを変更できます。

NIS パスワードおよび NIS+ パスワード

ネットワークで NIS を使用してユーザーの認証を行なっている場合、パスワード情報は NIS パスワードマップに保持されます。NIS では、パスワードの有効期間を指定できません。NIS パスワードマップに保持されているユーザーのパスワードを変更するには、コマンド `passwd -r nis` を使用します。

ネットワークで NIS+ を使用してユーザーの認証を行っている場合、パスワード情報は NIS+ データベースに保持されます。NIS+ データベース内の情報は、承認されたユーザーだけにアクセスを制限することによって保護できます。NIS+ データベースに保持されているユーザーのパスワードを変更するには、`passwd -r nisplus` コマンドを使用します。

LDAP パスワード

Oracle Solaris の LDAP ネームサービスは、パスワード情報とシャドウ情報を LDAP ディレクトリツリーの `ou=people` コンテナに格納します。Oracle Solaris LDAP ネームサービスクライアントでユーザーのパスワードを変更するには、`passwd -r ldap` コマンドを使用します。LDAP ネームサービスは、パスワードを LDAP リポジトリに格納します。

パスワードポリシーは、Sun Java System Directory Server 上で適用されます。つまり、クライアントの `pam_ldap` モジュールは、Sun Java System Directory Server で適用されているパスワードポリシー制御に従います。詳細は、『Solaris のシステム管理 (ネーミングとディレクトリサービス : DNS、NIS、LDAP 編)』の「LDAP ネームサービスのセキュリティーモデル」を参照してください。

パスワードの暗号化

パスワードの強力な暗号化は攻撃に対する最初の障壁になります。Oracle Solaris ソフトウェアには 6 つのパスワード暗号化アルゴリズムが提供されています。Blowfish、MD5、 および SHA アルゴリズムは、UNIX アルゴリズムよりも強力なパスワード暗号化を提供します。

パスワードアルゴリズムの識別子

サイトのアルゴリズムの構成は、`/etc/security/policy.conf` ファイルに指定します。`policy.conf` ファイルには、次の表に示す識別子でアルゴリズムを指定します。識別子とアルゴリズムのマッピングについては、`/etc/security/crypt.conf` ファイルを参照してください。

表 2–1 パスワードの暗号化アルゴリズム

識別子	説明	アルゴリズムのマニュアルページ
1	BSD システムや Linux システムの MD5 アルゴリズムと互換性のある MD5 アルゴリズム。	`crypt_bsdmd5(5)`
2a	BSD システムの Blowfish アルゴリズムと互換性のある Blowfish アルゴリズム。	`crypt_bsdbf(5)`
md5	BSD バージョンや Linux バージョンの MD5 よりも強力とされている Sun MD5 アルゴリズム。	`crypt_sunmd5(5)`
5	SHA256 アルゴリズム。SHA は、Secure Hash Algorithm (セキュアハッシュアルゴリズム) を表します。このアルゴリズムは、SHA-2 ファミリのメンバーです。SHA256 では 255 文字のパスワードがサポートされます。	`crypt_sha256(5)`
6	SHA512 アルゴリズム。	`crypt_sha512(5)`

表 2–1 パスワードの暗号化アルゴリズム (続き)

識別子	説明	アルゴリズムのマニュアルページ
__unix__	従来の UNIX 暗号化アルゴリズム。policy.conf ファイルのデフォルトモジュールはこのアルゴリズムです。	crypt_unix(5)

policy.conf ファイルのアルゴリズム構成

次に、policy.conf ファイルに指定されたデフォルトのアルゴリズム構成を示します。

```
#
...
# crypt(3c) Algorithms Configuration
#
# CRYPT_ALGORITHMS_ALLOW specifies the algorithms that are allowed to
# be used for new passwords.  This is enforced only in crypt_gensalt(3c).
#
CRYPT_ALGORITHMS_ALLOW=1,2a,md5,5,6

# To deprecate use of the traditional unix algorithm, uncomment below
# and change CRYPT_DEFAULT= to another algorithm.  For example,
# CRYPT_DEFAULT=1 for BSD/Linux MD5.
#
#CRYPT_ALGORITHMS_DEPRECATE=__unix__

# The Solaris default is the traditional UNIX algorithm.  This is not
# listed in crypt.conf(4) since it is internal to libc.  The reserved
# name __unix__ is used to refer to it.
#
CRYPT_DEFAULT=__unix__
...
```

CRYPT_DEFAULT の値を変更すると、新しいユーザーのパスワードは、新しい値に対応するアルゴリズムを使って暗号化されます。

既存のユーザーがパスワードを変更したときに新しいパスワードがどのアルゴリズムで暗号化されるかは、古いパスワードがどのように暗号化されているかによって異なります。たとえば、CRYPT_ALGORITHMS_ALLOW=1,2a,md5,5,6 かつ CRYPT_DEFAULT=1 であるとします。次の表は、パスワードの暗号化にどのアルゴリズムが使用されるかを示します。

識別子 = パスワードアルゴリズム		
元のパスワード	変更後のパスワード	意味
1 = crypt_bsdmd5	同じアルゴリズムを使用します。	1 識別子は CRYPT_DEFAULT の値でもあります。ユーザーのパスワードは引き続き crypt_bsdmd5 アルゴリズムで暗号化されます。

識別子=パスワードアルゴリズム		
元のパスワード	変更後のパスワード	意味
2a = crypt_bsdbf	同じアルゴリズムを使用します。	2a 識別子は CRYPT_ALGORITHMS_ALLOW リストにあります。このため、新しいパスワードは crypt_bsbdf アルゴリズムで暗号化されます。
md5 = crypt_md5	同じアルゴリズムを使用します。	md5 識別子は CRYPT_ALGORITHMS_ALLOW リストにあります。このため、新しいパスワードは crypt_md5 アルゴリズムで暗号化されます。
5 = crypt_sha256	同じアルゴリズムを使用します。	5 識別子は CRYPT_ALGORITHMS_ALLOW リストにあります。このため、新しいパスワードは crypt_sha256 アルゴリズムで暗号化されます。
6 = crypt_sha512	同じアルゴリズムを使用します。	6 識別子は CRYPT_ALGORITHMS_ALLOW リストにあります。このため、新しいパスワードは crypt_sha512 アルゴリズムで暗号化されます。
__unix__ = crypt_unix	crypt_bsdmd5 アルゴリズムを使用します。	__unix__ 識別子は CRYPT_ALGORITHMS_ALLOW リストにありません。このため、crypt_unix アルゴリズムを使用することはできません。新しいパスワードは CRYPT_DEFAULT アルゴリズムで暗号化されます。

選択したアルゴリズムの構成の詳細については、policy.conf(4) のマニュアルページを参照してください。パスワード暗号化アルゴリズムを指定する場合は、74 ページの「パスワードアルゴリズムの変更 (作業マップ)」を参照してください。

特殊なシステムアカウント

root アカウントは特殊なシステムアカウントの 1 つです。これらのアカウントのうち、root アカウントにのみパスワードが割り当てられ、ログインできます。nuucp アカウントはファイル転送用にログインできます。他のシステムアカウントは、ファイルを保護したり、または root の完全な権限を使用せずに管理プロセスを実行したりします。

注意 – システムアカウントのパスワード設定は決して変更しないでください。shadow ファイルの 2 番目のフィールドが NP または *LK*sys で送信されるアカウントは、システムアカウントを示します。

次の表に、一部のシステムアカウントとその使用方法の一覧を示します。システムアカウントは特殊な機能を実行します。各アカウントは、100 より小さい UID を持ちます。

表2–2 システムアカウントとその用途

システムアカウント	GID	用途
root	0	ほぼ無制限です。他の保護および許可より優先されます。root アカウントはシステム全体へのアクセス権を持ちます。root ログインのパスワードはきわめて厳密に保護する必要があります。root アカウント (スーパーユーザー) は、Oracle Solaris コマンドを所有します。
デーモン	1	バックグラウンド処理を制御します。
bin	2	一部の Oracle Solaris コマンドを所有します。
sys	3	多数のシステムファイルを所有します。
adm	4	一部のシステム管理ファイルを所有します。
lp	71	プリンタ用のオブジェクトデータファイルとスプールデータファイルを所有します。
uucp	5	UNIX 間のコピープログラム、UUCP 用のオブジェクトデータファイルとスプールデータファイルを所有します。
nuucp	9	システムにログインしてファイル転送を開始するために遠隔システムで使用されます。

遠隔ログイン

侵入者にとって、遠隔ログインは魅力的な手段です。Oracle Solaris は、リモートログインを監視、制限、および無効にする、いくつかのコマンドを提供します。手順については、66 ページの「ログインとパスワードの保護 (作業マップ)」を参照してください。

デフォルトでは、システムのマウスやキーボード、フレームバッファー、オーディオデバイスなど、特定のシステムデバイスを、遠隔ログインを通して制御したり読み取ったりすることはできません。詳細は、logindevperm(4) のマニュアルページを参照してください。

ダイヤルアップログイン

モデムやダイヤルアップポートを通してアクセスされうるコンピュータには、セキュリティー層をもう 1 つ追加できます。つまり、モデムやダイヤルアップポートを通してシステムにアクセスするユーザーには「ダイヤルアップパスワード」を要求することができます。ユーザーがシステムにアクセスできるようになるには、この追加パスワードを入力する必要があります。

ダイヤルアップパスワードを作成または変更できるのはスーパーユーザーのみです。システムの完全性を確保するために、月に一度はパスワードを変更する必要があります。この機能のもっとも有効な使用方法は、ゲートウェイシステムへのアク

セス権を取得するためのダイヤルアップパスワードを要求することです。ダイヤルアップパスワードの設定方法については、72 ページの「ダイヤルアップパスワードを作成する方法」を参照してください。

ダイヤルアップパスワードの作成には、/etc/dialups と /etc/d_passwd という 2 つのファイルが必要です。dialups ファイルには、ダイヤルアップパスワードを必要とするポートのリストを含みます。d_passwd ファイルには、追加のダイヤルアップパスワードとして暗号化パスワードを必要とするシェルプログラムのリストを含みます。これら 2 つのファイルの情報は次のように処理されます。

- /etc/passwd 内のユーザーのログインシェルが /etc/d_passwd 内のエントリと一致する場合、そのユーザーはダイヤルアップパスワードを入力する必要があります。
- /etc/passwd 内のユーザーのログインシェルが /etc/d_passwd 内で見つからない場合、そのユーザーはデフォルトのパスワードを入力する必要があります。デフォルトのパスワードは /usr/bin/sh のエントリです。
- /etc/passwd 内のログインシェルフィールドが空の場合、そのユーザーはデフォルトのパスワードを入力する必要があります。デフォルトのパスワードは /usr/bin/sh のエントリです。
- /etc/d_passwd に /usr/bin/sh のエントリがない場合、/etc/passwd 内のログインシェルフィールドが空のユーザー、または /etc/d_passwd 内のエントリと一致しないユーザーには、ダイヤルアップパスワードの入力を求めるプロンプトは表示されません。
- /etc/d_passwd ファイルのエントリが /usr/bin/sh:*: だけである場合、ダイヤルアップログインは無効です。

デバイスアクセスの制御

コンピュータシステムに接続された周辺機器は、セキュリティーリスクをもたらします。たとえば、マイクは会話をキャッチし、その会話を遠隔システムに送信します。CD-ROM の場合、その情報を CD-ROM に残して、CD-ROM デバイスを次に使うユーザーが読み取れるようにすることができます。プリンタは、遠隔サイトからもアクセスできます。システムの必須デバイスもまた、セキュリティー問題を引き起こす可能性があります。たとえば、hme0 のようなネットワークインタフェースは不可欠なデバイスとみなされています。

Oracle Solaris ソフトウェアには、デバイスアクセスを制御する方法が 2 つ用意されています。「デバイスポリシー」は、システムに不可欠なデバイスに対するアクセスの制限または防止を行うものです。デバイスポリシーはカーネル内で適用されます。「デバイス割り当て」は、周辺機器に対するアクセスの制限または防止を行う作業です。デバイス割り当ては、ユーザー割り当ての時点で適用されます。

デバイスポリシーは、特権を使用して、選択されたデバイスをカーネル内で保護します。たとえば、ネットワークインタフェースのデバイスポリシー (hme など) は、読み取りまたは書き込みに対してすべての権限を要求します。

デバイス割り当ては、承認を利用してプリンタやマイクなどの周辺機器を保護します。デフォルトでは、デバイス割り当ては無効な状態になっています。デバイス割り当てが有効な状態では、この構成を変更してデバイスの使用を防いだり、デバイスアクセスに承認が必要なように設定したりできます。デバイスの使用割り当てが行われると、現在のユーザーがその割り当てを解除するまでほかのユーザーはそのデバイスにアクセスできません。

デバイスアクセスを制御する手段として、次に示すようないくつかの方法で Oracle Solaris システムを構成できます。

- デバイスポリシーを設定する - Oracle Solaris では、特定のデバイスにプロセスがアクセスする場合、そのプロセスが特定の権限のもとで実行されていることを必要条件とすることができます。それらの権限を持たないプロセスは、そのデバイスを使用できません。起動時に、Oracle Solaris ソフトウェアはデバイスポリシーを設定します。Sun 以外のドライバは、そのインストール時にデバイスポリシーを組み込むことができます。インストール後、管理者はデバイスポリシーをデバイスに追加できます。
- デバイスを割り当て可能にする – デバイス割り当てを有効にする際には、一度に 1 人しかそのデバイスを使用できないように制限できます。また、ユーザーが何らかのセキュリティー要件を満たすように要求することも可能です。たとえば、そのデバイスを使用する承認を得ていることを要求できます。
- デバイスの使用を防ぐ – コンピュータシステム上のどのユーザーも特定のデバイス (マイクなど) を使用できないように設定できます。特定のデバイスを使用できない状態にする例としては、コンピュータキオスクが挙げられます。
- デバイスを特定のゾーンに限定する – デバイスの使用を非大域ゾーンにだけ割り当てることができます。詳細は、『Oracle Solaris のシステム管理 (Oracle Solaris コンテナ : 資源管理と Oracle Solaris ゾーン)』の「非大域ゾーンでのデバイスの使用」を参照してください。デバイスとゾーンに関する一般的な説明は、『Oracle Solaris のシステム管理 (Oracle Solaris コンテナ : 資源管理と Oracle Solaris ゾーン)』の「ゾーンで構成されるデバイス」を参照してください。

デバイスポリシー (概要)

デバイスポリシーメカニズムを使用することで、デバイスを開こうとするプロセスに特定の権限を要求するように指定できます。デバイスポリシーによって保護されたデバイスをアクセスできるのは、デバイスポリシーで指定されている権限で稼働しているプロセスだけです。Oracle Solaris はデフォルトのデバイスポリシーを提供します。たとえば、`hme0` などのネットワークインタフェースは、インタフェースにア

クセスするプロセスが net_rawaccess 権限で稼働していることを必要とします。この要件はカーネルで適用されます。権限の詳細は、199 ページの「特権 (概要)」を参照してください。

以前のリリースでは、デバイスノードの保護はファイルアクセス権だけで行われました。たとえば、グループ sys が所有しているデバイスをオープンできるのはこのグループのメンバーだけでした。デバイスを開くことができるユーザーをアクセス権が予測することはありません。デバイスは、ファイルアクセス権によって保護されるとともに、デバイスポリシーでも保護されます。たとえば、/dev/ip ファイルのアクセス権は 666 です。しかし、このデバイスは適切な権限を持つプロセスによってしかオープンできません。

デバイスポリシーの設定は監査の対象とすることができます。デバイスポリシーの変更は、AUE_MODDEVPLCY 監査イベントによって記録されます。

デバイスポリシーの詳細は、次のページを参照してください。

- 86 ページの「デバイスポリシーの設定 (作業マップ)」
- 100 ページの「デバイスポリシーコマンド」
- 207 ページの「特権とデバイス」

デバイス割り当て (概要)

デバイス割り当てメカニズムを使用すれば、CD-ROM などの周辺機器に対するアクセスを制限できます。このメカニズムは、システム管理者によってローカルに管理されます。デバイス割り当てが有効になっていない場合、周辺機器の保護はファイルアクセス権によってのみ行われます。たとえば、デフォルトでは周辺機器は次のように使用できます。

- すべてのユーザーがフロッピーディスクや CD-ROM の読み取りと書き込みが行えます。
- すべてのユーザーがマイクを接続できます。
- すべてのユーザーが接続されたプリンタにアクセスできます。

デバイス割り当てを行うことで、承認されたユーザーにだけデバイスの使用を限定できます。デバイス割り当てによって、デバイスアクセスを完全に防ぐこともできます。デバイスを割り当てるユーザーは、そのユーザー自身が割り当てを解除するまでそのデバイスを独占的に使用できます。デバイスの割り当てが解除される際には、残っているすべてのデータがデバイスクリーンスクリプトによって消去されます。デバイスにスクリプトがない場合には、デバイスクリーンスクリプトを作成してそのデバイスから情報を一掃できます。この例は、107 ページの「新しいデバイスクリーンスクリプトの作成」を参照してください。

デバイス割り当てに関連した試み (デバイスの割り当て、デバイスの割り当て解除、割り当て可能なデバイスの一覧表示) は、監査の対象とすることができます。監査イベントは、other 監査クラスの一部です。

デバイス割り当ての詳細は、次のページを参照してください。

- 89 ページの「デバイス割り当ての管理 (作業マップ)」
- 100 ページの「デバイスの割り当て」
- 101 ページの「デバイス割り当てコマンド」

マシンリソースへのアクセス制御

システム管理者は、システム活動の制御や監視を行うことができます。システム管理者は、だれがどのリソースを使用できるかを制限したり、リソースの使用状況を記録したり、だれがリソースを使用しているかを監視したりできます。システム管理者は、リソースの不適切な使用を最小限に抑えるようにシステムを設定することもできます。

スーパーユーザーの制限と監視

システムでスーパーユーザーアクセスを行うには、root パスワードが必要です。デフォルトの構成では、ユーザーは遠隔からシステムに root としてログインできません。遠隔ログインするとき、ユーザーは自分のユーザー名でログインしてから、su コマンドを使用して root になる必要があります。管理者は、必要に応じて su コマンドを使用中のユーザー (特にスーパーユーザーのアクセス権を取得しようとしているユーザー) を監視できます。スーパーユーザーを監視したり、スーパーユーザーのアクセス権を制限したりする手順については、78 ページの「スーパーユーザーの監視と制限 (作業マップ)」を参照してください。

役割によるアクセス制御を構成してスーパーユーザーを置き換える

役割によるアクセス制御 (RBAC) は、スーパーユーザーの権限を制限することを目的として設計されています。スーパーユーザーすなわち root ユーザーは、システムのすべてのリソースにアクセスできますが、RBAC を使用すれば、スーパーユーザーの権限を個別の権限からなる役割の集合に置き換えることができます。たとえば、ユーザーアカウントの作成を行う役割やシステムファイルの変更を行う役割を個別に設定できます。特定の機能または機能群を扱う役割を設定したら、これらの機能を root の機能から取り除くことができます。

個々の役割を引き受けるためには、既存のユーザーが自分のユーザー名とパスワードを使ってログインする必要があります。ログインしたユーザーは、特別な役割パスワードを入力してその役割を引き受けます。これによって、他人が root パスワードを知ったとしても、システムに損傷を与える能力は限定されます。RBAC の詳細は、188 ページの「役割によるアクセス制御 (概要)」を参照してください。

マシンリソースの意図しない使用の防止

システム管理者は、自分自身やユーザーによって意図しないエラーが引き起こされないように防止できます。

- たとえば、PATH 変数を正しく設定することによって、トロイの木馬の実行を防止できます。
- 制限されたシェルをユーザーに割り当てることもできます。システムのうち各人の作業に必要な部分だけをユーザーに提供するという方法でシェル機能を制限すると、ユーザーエラーを避けることができます。実際、慎重に設定すれば、作業を能率的に行う上で必要な部分以外にユーザーがアクセスできないように制限できます。
- そのユーザーがアクセスする必要がないファイルには、限定的なアクセス権を設定できます。

PATH 変数の設定

PATH 変数を正しく設定しないと、他人が持ち込んだプログラムを誤って実行し、データを壊したりシステムを損傷したりするおそれがあります。このようなプログラムはセキュリティー上の危険を招くので、「トロイの木馬」と呼ばれます。たとえば、公開ディレクトリの中に別の su プログラムが置かれていると、システム管理者が気づかずに実行してしまう可能性があります。このようなスクリプトは正規の su コマンドとまったく同じに見えます。このようなスクリプトは実行後に自らを削除してしまうため、トロイの木馬が実際に実行されたという証拠はほとんど残りません。

PATH 変数はログイン時に自動的に設定されます。パスは、起動ファイル、すなわち `.login`、 `.profile`、 および `.cshrc` を通して設定されます。現在のディレクトリ (.) への検索パスを最後に指定すれば、トロイの木馬のようなタイプのプログラムを実行するのを防ぐことができます。スーパーユーザーの PATH 変数には、現在のディレクトリを指定しないでください。

自動セキュリティー拡張ツール (ASET) は、起動ファイルの PATH 変数が正しく設定されているかどうかを調べます。また、PATH 変数にドット (.) エントリが含まれていないか確認します。

ユーザーに制限付きシェルを割り当てる

標準シェルを使用すると、ユーザーはファイルを開く、コマンドを実行するなどの操作を行うことができます。制限付きシェルを使用すると、ディレクトリの変更やコマンドの実行などのユーザー能力を制限できます。制限付きシェルは、`/usr/lib/rsh` コマンドで呼び出されます。制限付きシェルは、遠隔シェル `/usr/sbin/rsh` ではありません。

標準のシェルと異なる点は次のとおりです。

- ユーザーはホームディレクトリ内に限定されるため、 cd コマンドを使用してディレクトリを変更できません。したがって、システムファイルを閲覧することはできません。
- ユーザーは PATH 変数を変更できないため、システム管理者によって設定されたパスのコマンドしか使用できません。さらに、完全なパス名を使ってコマンドやスクリプトを実行することもできません。
- ユーザーは、> または >> を使用して出力をリダイレクトできません。

制限付きシェルでは、ユーザーが使用できるシステムファイルを制限できます。このシェルは、特定のタスクを実行するユーザーのために限られた環境を作成します。ただし、制限付きシェルは完全に安全なわけではありません。このシェルの目的は、あくまでも、経験の少ないユーザーが誤ってシステムファイルを損傷するのを防止することです。

制限付きシェルについては、rsh(1M) のマニュアルページを参照してください。このマニュアルページは、man -s1m rsh コマンドで表示できます。

ファイル内のデータへのアクセス制限

Oracle Solaris はマルチユーザー環境なので、ファイルシステムのセキュリティーは、システムのもっとも基本的なセキュリティーリスクです。ファイルの保護には、従来の UNIX のファイル保護と、より確実なアクセス制御リスト (ACL) との両方が使用できます。

一部のユーザーには特定のファイルの読み取りを許可し、別のユーザーには特定のファイルを変更または削除するアクセス権を与えることができます。一方、あるデータを、どのユーザーからも読み取られないよう設定することもできます。ファイルのアクセス権の設定方法については、第 6 章「ファイルアクセスの制御 (作業)」を参照してください。

setuid 実行可能ファイルの制限

実行可能ファイルがセキュリティーリスクとなる場合があります。多くの実行可能プログラムは、スーパーユーザー (root) として実行しなければ適切に動作しません。これらの setuid プログラムは、ユーザー ID が 0 に設定された状態で実行されます。このようなプログラムはだれが実行したとしても root ID で実行されます。root ID で動作するプログラムは、プログラムがセキュリティーを念頭に置いて作成されていない限り、セキュリティーの問題をはらんでいます。

Oracle が setuid ビットを root に設定して出荷する実行可能プログラムを除き、setuid プログラムの使用を許可するべきではありません。setuid プログラムの使用を禁止できない場合は、その使用を制限する必要があります。しっかりした管理を行うためには setuid プログラムの数を少なくする必要があります。

詳細は、141 ページの「実行可能ファイルを原因とするセキュリティーへの悪影響を防止する」を参照してください。作業手順については、154 ページの「セキュリティーリスクのあるプログラムからの保護 (作業マップ)」を参照してください。

自動セキュリティー拡張ツールの使用

ASET セキュリティーパッケージには、システムのセキュリティーを制御して監視できるように、自動管理ツールが組み込まれています。ASET には、 低、中、高という 3 つのセキュリティーレベルがあります。システム管理者は ASET セキュリティーレベルを指定します。上のレベルほど、ASET のファイル制御機能が増え、ファイルアクセス は減少し、システムセキュリティーが厳しくなります。詳細は、第 7 章「自動セキュリティー拡張ツールの使用 (手順)」を参照してください。

Oracle Solaris Security Toolkit の使用

ASET がシステムにセキュリティー上のわずかな変更を加えるために使用できるのに対して、Oracle Solaris Security Toolkit は Oracle Solaris システムの最小化、強化、保護などに対応できる、柔軟で拡張性のあるメカニズムを備えています。Oracle Solaris Security Toolkit は、非公式には JASS ツールキットと呼ばれ、システムにセキュリティー変更を加えることができます。このツールを使用して、Oracle Solaris システムのセキュリティー状態に関するレポートを生成できます。このツールには、ツールの直前の実行を取り消す機能もあります。JASS ツールキットは、Oracle and Sun (`http://www.oracle.com/us/sun/index.htm`) からダウンロードできます。「Sun Downloads: A-Z listing」をクリックして、アルファベット順のダウンロードの一覧で文字列 `Solaris Security Toolkit` を検索します。

このツールキットの詳細は、『Securing Systems with the Solaris Security Toolkit』(Alex Noordergraaf、 Glenn Brunette 共著、ISBN 0-13-141071-7、 2003 年 6 月発行) を参照してください。このドキュメントは、Sun Microsystems Press によって発行されている Sun BluePrints Series の 1 つです。

デフォルトでのセキュリティー強化 (Secure By Default) 構成の使用

デフォルトでは、Solaris 10 リリースのインストール時に多数のネットワークサービスが有効になります。システムのネットワーク接続を制限するには、`netservices limited` コマンドを実行します。このコマンドは、「デフォルトでのセキュリティー強化 (Secure By Default)」(SBD) 構成を有効にします。SBD により、ネットワーク要求を受け付けるネットワークサービスは `sshd` デーモンだけになります。ほかのネットワークサービスはすべて無効になるか、ローカル要求だけを処理するよ

うになります。ftp などの個々のネットワークサービスを有効にするには、サービス管理機能 (SMF) を使用します。詳細は、netservices(1M) および smf(5) のマニュアルページを参照してください。

リソース管理機能の使用

Oracle Solaris ソフトウェアには、精巧なリソース管理機能があります。これらの機能を使用することで、サーバー統合環境内のアプリケーションによるリソース利用の割り当て、スケジュール、監視、上限設定などを行うことができます。リソース制御フレームワークにより、プロセスが使用するシステムリソースを制限できます。このような制約を行うことで、システムリソースを混乱させようとするスクリプトによるサービス拒否攻撃を防ぎやすくなります。

Oracle Solaris リソース管理機能により、特定のプロジェクトに必要なリソースを割り当てられます。また、使用できるリソースを動的に調整することもできます。詳細は、『Oracle Solaris のシステム管理 (Oracle Solaris コンテナ : 資源管理と Oracle Solaris ゾーン)』のパート I「資源管理」を参照してください。

Oracle Solaris ゾーンの使用

Oracle Solaris ゾーンは、単一の Oracle Solaris OS インスタンス内に存在するほかのシステムからプロセスが分離されるアプリケーション実行環境です。この分離を行うことで、1 つのゾーン内で稼働しているプロセスがほかのゾーンで稼働しているプロセスを監視したりそれらのプロセスに影響を及ぼしたりすることが防止されます。これは、スーパーユーザー権限によって稼働しているプロセスでも同様です。

Oracle Solaris ゾーンは、単一のサーバー上にアプリケーションを複数配置する環境に適しています。詳細は、『Oracle Solaris のシステム管理 (Oracle Solaris コンテナ : 資源管理と Oracle Solaris ゾーン)』のパート II「ゾーン」を参照してください。

マシンリソースの使用状況の監視

システム管理者は、システムの動作を監視する必要があります。次の点を含め、マシンのあらゆる側面に注意する必要があります。

- 通常の負荷はどの程度か
- 誰がシステムへのアクセス権を持っているか
- 各ユーザーはいつシステムにアクセスするか
- システムでは通常どのようなプログラムを実行するか

このような情報を把握していれば、ツールを使用してシステムの使用状況を監査し、各ユーザーのアクティビティーを監視できます。セキュリティー侵害と思われる場合は、監視作業が特に役立ちます。監査サービスの詳細は、第 28 章「Oracle Solaris 監査 (概要)」を参照してください。

ファイルの整合性の監視

システム管理者は、管理対象のシステムにインストールされたファイルが予想外の方法で変更されないように確実な対策をとる必要があります。大規模インストールでは、各システム上のソフトウェアスタックの比較や報告を行うツールを使用すればシステムの追跡、記録が行えます。基本監査報告機能 (BART) を使用すると、一定期間にわたって1 つ以上のシステムをファイルレベルでチェックし、システムを包括的に検証できます。一定期間にわたってすべてのシステムまたは 1 つのシステムにおける BART 目録の変化を調べることで、システムの整合性を検証できます。BART には、目録作成機能、目録比較機能、レポート生成規則などが用意されています。詳細は、第 5 章「基本監査報告機能の使用方法 (作業)」を参照してください。

ファイルアクセスの制御

Oracle Solaris はマルチユーザー環境です。マルチユーザー環境では、システムにログインしているすべてのユーザーが、ほかのユーザーに属しているファイルを読み取ることができます。さらに、適切なアクセス権をもっているユーザーは、ほかのユーザーに属しているファイルを使用できます。詳細は、第 6 章「ファイルアクセスの制御 (作業)」を参照してください。ファイルに適切なアクセス権を設定するステップごとの手順については、142 ページの「ファイルの保護 (作業マップ)」を参照してください。

暗号化によるファイルの保護

ほかのユーザーがアクセスできないようにすることによって、ファイルを安全に保つことができます。たとえば、アクセス権 `600` の付いたファイルに、所有者やスーパーユーザー以外の人がアクセスすることはできません。アクセス権 `700` の付いたディレクトリも同様です。ただし、ほかのだれかがユーザーパスワードや `root` パスワードを推測して発見すると、そのファイルにアクセスできます。さらに、アクセス不能なはずのファイルも、システムファイルのバックアップをオフラインメディアにとるたびに、バックアップテープ上に保存されます。

暗号化フレームワークには、ファイルを保護するコマンドとして、`digest`、 `mac`、 および `encrypt` コマンドが用意されています。詳細は、第 13 章「Oracle Solaris の暗号化フレームワーク (概要)」を参照してください。

アクセス制御リストの使用

ACL (「アクル」と読む) では、ファイルアクセス権の制御をより強化できます。ACL は、従来の UNIX ファイル保護機能では不十分な場合に追加で使用します。従来の UNIX ファイル保護機能は、 所有者、グループ、その他のユーザーという 3 つのユーザークラスに読み取り権、書き込み権、実行権を提供します。ACL では、ファイルセキュリティーを管理するレベルがさらに詳細になります。

ACL では、次のファイルアクセス権を定義できます。

- 所有者のファイルアクセス権
- 所有者のグループのファイルアクセス権
- 所有者のグループに属していないユーザーのファイルアクセス権
- 特定ユーザーのファイルアクセス権
- 特定グループのファイルアクセス権
- 以上のカテゴリそれぞれのデフォルトアクセス権

ACL の使用についての詳細は、138 ページの「アクセス制御リストによる UFS ファイルの保護」を参照してください。

マシン間でのファイルの共有

ネットワークファイルサーバーは、どのファイルを共有できるかを制御できます。また、共有ファイルにアクセスできるクライアント、およびそれらのクライアントに許可するアクセス権の種類も制御します。一般に、ファイルサーバーは、すべてのクライアントまたは特定のクライアントに、読み取り権と書き込み権、または読み取り専用アクセス権を与えることができます。アクセス制御は、`share` コマンドでリソースを利用可能にするときに指定します。

ファイルサーバー上の `/etc/dfs/dfstab` ファイルは、サーバーがネットワーク上のクライアントに提供するファイルシステムをリストします。ファイルシステムの共有の詳細については、『Solaris のシステム管理 (ネットワークサービス)』の「ファイルシステムの自動共有」を参照してください。

ZFS ファイルシステムの NFS 共有を作成すると、共有を削除するまでファイルシステムは永続的に共有されます。システムを再起動すると、SMF は共有を自動的に管理します。詳細は、『Oracle Solaris ZFS 管理ガイド』の第 3 章「Oracle Solaris ZFS ファイルシステムと従来のファイルシステムの相違点」を参照してください。

共有ファイルへの **root** アクセスの制限

一般的にスーパーユーザーは、ネットワーク上で共有されるファイルシステムには `root` としてアクセスできません。NFS システムは、要求者のユーザーをユーザー ID `60001` を持つユーザー `nobody` に変更することによって、マウントされているファイルシステムへの `root` アクセスを防止します。ユーザー `nobody` のアクセス権は、公共ユーザーに与えられているアクセス権と同じです。つまり、ユーザー `nobody` のアクセス権は資格をもたないユーザーのものと同じです。たとえば、ファイルの実行権しか公共に許可していなければ、ユーザー `nobody` はそのファイルを実行することしかできません。

NFS サーバーは、共有ファイルシステムのスーパーユーザー能力をホスト単位で与えることができます。この権限を与えるには、`share` コマンドの `root=`*hostname* オプ

ションを使用します。このオプションは慎重に使用してください。NFS のセキュリティーオプションの詳細は、『Solaris のシステム管理 (ネットワークサービス)』の第 6 章「ネットワークファイルシステムへのアクセス (リファレンス)」を参照してください。

ネットワークアクセスの制御

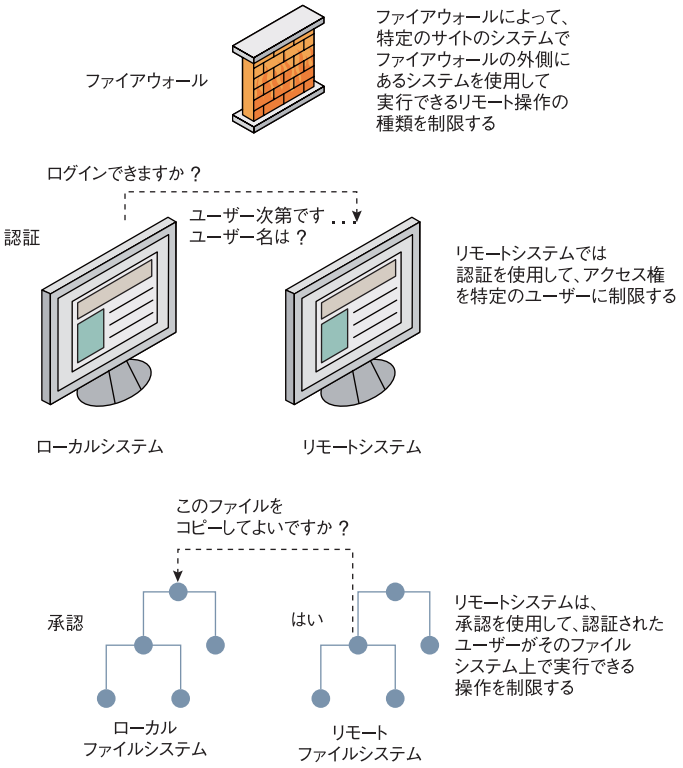
コンピュータは通常、コンピュータのネットワークの一部です。ネットワークでは、接続されたコンピュータの間で情報を交換できます。さらに、ネットワークに接続されたコンピュータは、ネットワーク上のほかのコンピュータにあるデータなどのリソースにアクセスできます。コンピュータをネットワーク化すると処理能力と性能が向上しますが、同時にセキュリティーの複雑化にもつながります。

たとえば、コンピュータのネットワーク内では、個々のシステムは情報を共有できます。未承認アクセスがセキュリティーリスクとなります。多くの人々がネットワークにアクセスするので、(特にユーザーエラーを通して) 未承認アクセスが発生する可能性も大きくなります。また、パスワードの不適切な扱いも未承認アクセスの原因となります。

ネットワークセキュリティー機構

一般にネットワークセキュリティーは、遠隔システムからの操作の制限またはブロックを通して維持されます。次の図は、遠隔操作に適用できるセキュリティー制限を示します。

図 2–1 遠隔操作のセキュリティー制限



遠隔アクセスの認証と承認

「認証」とは、遠隔システムにアクセスできるユーザーを特定のユーザーに限定する方法です。認証は、システムレベルでもネットワークレベルでも設定できます。ユーザーが遠隔システムにアクセスすると、「承認」という方法でそのユーザーが実行できる操作が制限されます。次の表は、認証と承認を提供するサービスを示したものです。

表 2–3 遠隔アクセス用の認証サービスと承認サービス

サービス	説明	詳細
IPsec	IPsec は、ホストに基づく認証および認可に基づく認証と、ネットワークトラフィックの暗号化を行います。	『Solaris のシステム管理 (IP サービス)』の第 19 章「IP セキュリティーアーキテクチャー (概要)」
Kerberos	Kerberos は、システムにログインしているユーザーの認証と承認を暗号化を通して行います。	この例は、402 ページの「Kerberos サービスの動作」を参照してください。

表 2–3 遠隔アクセス用の認証サービスと承認サービス (続き)

サービス	説明	詳細
LDAP と NIS+	LDAP ディレクトリサービスと NIS+ ネームサービスは、共にネットワークレベルで認証および承認を行うことができます。	『Solaris のシステム管理 (ネーミングとディレクトリサービス : DNS、NIS、LDAP 編)』 および『Solaris のシステム管理 (ネーミングとディレクトリサービス : NIS+ 編)』
遠隔ログインコマンド	遠隔ログインコマンドを使用すると、ユーザーはネットワーク経由で遠隔システムにログインし、そのリソースを使用できます。遠隔ログインコマンドには `rlogin`、 `rcp`、 `ftp` などがあります。「信頼される (trusted) ホスト」の場合、認証は自動的に処理されます。それ以外の場合は、自分自身を認証するように求められます。	『Solaris のシステム管理 (ネットワークサービス)』の第 29 章「リモートシステムへのアクセス (手順)」
SASL	簡易認証セキュリティー層 (SASL) は、ネットワークプロトコルに認証サービスとセキュリティーサービス (オプション) を提供するフレームワークです。プラグインによって、適切な認証プロトコルを選択できます。	357 ページの「SASL (概要)」
Secure RPC	Secure RPC を使用すると、遠隔マシン上で要求を出したユーザーの認証が行われ、ネットワーク環境のセキュリティーが高まります。Secure RPC には、UNIX、 DES、 または Kerberos 認証システムを使用できます。	331 ページの「Secure RPC の概要」
	Secure RPC を使用すると、NFS 環境にセキュリティーを追加できます。Secure RPC を備えた NFS 環境を Secure NFS と呼びます。Secure NFS は、公開鍵に Diffie-Hellman 認証を使用します。	331 ページの「NFS サービスと Secure RPC」
Secure Shell	Secure Shell は、セキュリティー保護されていないネットワークを介したネットワークトラフィックを暗号化します。Secure Shell は、パスワードまたは公開鍵、あるいはこの両方による認証を提供します。Secure Shell は、公開鍵に RSA および DSA 認証を使用します。	361 ページの「Oracle Solaris Secure Shell (概要)」

Secure RPC に匹敵する機能として、Oracle Solaris の「特権ポート」メカニズムがあります。特権ポートには、`1024` 未満のポート番号が割り当てられます。クライアントシステムは、クライアントの資格を認証したあと、特権ポートを使用してサーバーへの接続を設定します。次に、サーバーは接続のポート番号を検査してクライアントの資格を検証します。

Oracle Solaris ソフトウェアを使用していないクライアントは、特権ポートを使用して通信できないことがあります。クライアントが特権ポートを使って通信できない場合は、次のようなエラーメッセージが表示されます。

```
“Weak Authentication
NFS request from unprivileged port”
```

ファイアウォールシステム

ファイアウォールシステムを設定すると、ネットワーク内のリソースを外部のアクセスから保護できます。「ファイアウォールシステム」は、内部ネットワークと外部ネットワークの間のバリアとして機能するセキュリティー保護ホストです。内部ネットワークは、ほかのネットワークを「信頼できる状態でない」ものとして扱います。 内部ネットワークと、インターネットなどの外部ネットワークとの間に、このような設定を必ず行うようにしてください。

ファイアウォールはゲートウェイとしても機能しますし、バリアとしても機能します。ファイアウォールは、まず、ネットワーク間でデータを渡すゲートウェイとして機能します。さらに、ファイアウォールは、データが勝手にネットワークに出入りしないようにブロックするバリアとして機能します。ファイアウォールは、内部ネットワーク上のユーザーに対して、ファイアウォールシステムにログインして遠隔ネットワーク上のホストにアクセスするように要求します。また、外部ネットワーク上のユーザーは、内部ネットワーク上のホストにアクセスする前に、まずファイアウォールシステムにログインしなければなりません。

ファイアウォールは、一部の内部ネットワーク間でも有効です。たとえば、ファイアウォール、すなわちセキュリティー保護ゲートウェイコンピュータを設定することによって、パケットの転送を制限できます。ゲートウェイコンピュータは、ゲートウェイ自身をパケットの発信元アドレスまたは着信先アドレスとしないような、2 つのネットワーク間のパケット交換を禁止できます。また、ファイアウォールは、特定のプロトコルについてのみパケットを転送するように設定する必要があります。たとえば、パケットでメールを転送できるが、`telnet` や `rlogin` コマンドのパケットは転送できないようにできます。ASET は、高度なセキュリティーを適用して実行すると、インターネットプロトコル (IP) パケットの転送機能を無効にします。

さらに、内部ネットワークから送信されるすべての電子メールは、まずファイアウォールシステムに送信されます。ファイアウォールは、このメールを外部ネットワーク上のホストに転送します。ファイアウォールシステムは、すべての着信電子メールを受信して内部ネットワーク上のホストに配信するという役割も果たします。

注意 – ファイアウォールは、アクセス権のないユーザーが内部ネットワーク上のホストにアクセスする行為を防止します。ファイアウォールに適用される厳密で確実なセキュリティーを管理する必要がありますが、ネットワーク上の他のホストのセキュリティーはもっと緩やかでも構いません。ただし、ファイアウォールシステムを突破できる侵入者は、内部ネットワーク上の他のすべてのホストへのアクセスを取得できる可能性があります。

ファイアウォールシステムには、信頼されるホストを配置しないでください。「信頼されるホスト」とは、ユーザーがログインするときに、パスワードを入力する必要がないホストのことです。ファイアウォールシステムでは、ファイルシステムを共有しないでください。また、ほかのサーバーのファイルシステムをマウントしないでください。

次の技術は、システムを強化してファイアウォールを確立するのに利用できます。

- ASET は、ファイアウォールシステムにおける高セキュリティーを実現します。詳細は、第 7 章「自動セキュリティー拡張ツールの使用 (手順)」を参照してください。
- Oracle Solaris Security Toolkit は、非公式には JASS ツールキットと呼ばれ、Oracle Solaris システムを強化してファイアウォールを確立できます。このツールキットは、Oracle Sun (`http://www.oracle.com/us/sun/index.htm`) の Sun Downloads Web サイトからダウンロードできます。
- IPsec と Oracle Solaris IP フィルタには、ファイアウォール保護機能があります。ネットワークトラフィックの保護についての詳細は、『Solaris のシステム管理 (IP サービス)』のパート IV「IP セキュリティー」を参照してください。

暗号化システムとファイアウォールシステム

ほとんどのローカルエリアネットワークでは、データは「パケット」と呼ばれるブロック単位でコンピュータ間で転送されます。ネットワークの外部にいるアクセス権のないユーザーが、「パケットスマッシング」という方法により、データを変更または破壊する可能性があります。

パケットスマッシングでは、パケットが宛先に到達する前に取り込まれます。侵入者は、その内容に勝手なデータを書き込み、パケットを元のコースに戻します。ローカルエリアネットワーク上では、パケットはサーバーを含むすべてのシステムに同時に到達するので、パケットスマッシングは不可能です。ただし、ゲートウェイ上ではパケットスマッシングが可能なため、ネットワーク上のすべてのゲートウェイを保護する必要があります。

もっとも危険なのは、データの完全性に影響するような攻撃です。このような攻撃を受けると、パケットの内容が変更されたり、ユーザーが偽装されたりします。盗聴を伴う攻撃では、データの完全性が損なわれることはありません。盗聴者は、会

話を記録して、あとで再生します。盗聴者がユーザーを偽装することはありません。盗聴攻撃によってデータの完全性が損なわれることはありませんが、プライバシが侵害されます。ネットワーク上でやりとりされるデータを暗号化すると、重要な情報のプライバシを保護できます。

- セキュリティー保護されていないネットワークを介した遠隔操作を暗号化する場合は、第 19 章「Oracle Solaris Secure Shell の使用 (手順)」を参照してください。
- ネットワークを介したデータの暗号化と認証を行う場合は、第 21 章「Kerberos サービスについて」を参照してください。
- IP データグラムを暗号化する場合は、『Solaris のシステム管理 (IP サービス)』の第 19 章「IP セキュリティーアーキテクチャー (概要)」を参照してください。

セキュリティー問題の報告

セキュリティーの問題が発生した可能性がある場合は、Computer Emergency Response Team/Coordination Center (CERT/CC) に連絡してください。CERT/CC は、Defense Advanced Research Projects Agency (DARPA) の資金提供を受けたプロジェクトで、カーネギメロン大学の Software Engineering Institute にあります。CERT/CC はセキュリティー問題の解決を支援できます。また、特定のニーズに合った他の Computer Emergency Response Team を紹介することもできます。最新の連絡先情報については、CERT/CC (`http://www.cert.org/contact_cert/`) Web サイトを参照してください。

◆ ◆ ◆ 第 3 章

3 システムアクセスの制御 (作業)

この章では、Oracle Solaris システムにアクセスできるユーザーを制御する方法について説明します。この章の内容は次のとおりです。

システムセキュリティーの概要については、第 2 章「マシンセキュリティーの管理 (概要)」を参照してください。

システムアクセスの制御 (作業マップ)

コンピュータのセキュリティーレベルは、コンピュータのもっとも弱い点によって決まります。次の作業マップは、どのような分野を監視してそのセキュリティーを確保すべきかを示しています。

作業	説明	参照先
ユーザーログインを監視、許可、および拒否します	一般的でないログインアクティビティーを監視します。ログインを一時的に禁止します。ダイヤルアップログインを管理します。	66 ページの「ログインとパスワードの保護 (作業マップ)」
パスワードの強力な暗号化を可能にします	ユーザーパスワードを暗号化するアルゴリズムを指定します。ほかのアルゴリズムをインストールします。	74 ページの「パスワードアルゴリズムの変更 (作業マップ)」
スーパーユーザーによる処理の監視と制限を行います	スーパーユーザーによる処理を定期的に監視します。root ユーザーによる遠隔ログインを禁止します。	78 ページの「スーパーユーザーの監視と制限 (作業マップ)」

作業	説明	参照先
ハードウェア設定へのアクセスを禁止します	通常のユーザーを PROM にアクセスさせません。	81 ページの「SPARC: システムハードウェアに対するアクセスの制御 (作業マップ)」

ログインとパスワードの保護 (作業マップ)

次の作業マップは、ユーザーログインを監視する手順と、ユーザーログインを無効にする手順を示しています。

作業	説明	参照先
ユーザーのログイン状態を表示します	ユーザーのログインアカウントについての広範な情報 (フルネーム、パスワードの有効期限など) を表示します。	67 ページの「ユーザーのログイン状態を表示する方法」
パスワードを所有していないユーザーを発見します	パスワードを必要としないアカウントを持つユーザーだけを検出します。	68 ページの「パスワードを持たないユーザーを表示する方法」
ログインを一時的に無効にします	システムシャットダウンや定常的な保守の中でマシンへのユーザーログインを拒否します。	68 ページの「ユーザーのログインを一時的に無効にする方法」
ログイン失敗操作を保存します	正しいパスワードの入力に 5 回失敗したユーザーのログを作成します。	69 ページの「ログイン失敗操作を監視する方法」
失敗したすべてのログイン操作を保存します	失敗したログイン操作のログを作成します。	70 ページの「すべてのログイン失敗操作を監視する方法」
ダイヤルアップパスワードを作成します	モデムやダイヤルアップポートを通して遠隔ログインするユーザーに追加パスワードを要求します。	72 ページの「ダイヤルアップパスワードを作成する方法」
ダイヤルアップログインを一時的に無効にします	ユーザーがモデムやポートを通して遠隔からダイヤルできないようにします。	73 ページの「ダイヤルアップログインを一時的に無効にする方法」

ログインとパスワードのセキュリティー

遠隔ログインを制限し、ユーザーにパスワードを要求するようにできます。失敗したアクセス操作を監視し、ログインを一時的に無効にすることもできます。

▼ ユーザーのログイン状態を表示する方法

1 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 **logins** コマンドを使用してユーザーのログイン状態を表示します。

```
# logins -x -l username
```

`-x` ログイン状態情報の拡張セットを表示します。

`-l` *username* 指定するユーザーのログイン状態を表示します。変数 *username* はユーザーのログイン名です。複数のログイン名は、コンマで区切って指定します。

`logins` コマンドは、適切なパスワードデータベースを使ってユーザーのログイン状態を表示します。このデータベースは、ローカルの `/etc/passwd` ファイルか、ネームサービスのパスワードデータベースです。詳細は、`logins(1M)` のマニュアルページを参照してください。

例 3–1 ユーザーのログイン状態を表示する

次の例では、ユーザー `rimmer` のログイン状態が表示されます。

```
# logins -x -l rimmer
rimmer       500     staff           10   Annalee J. Rimmer
                     /export/home/rimmer
                     /bin/sh
                     PS 010103 10 7 -1
```

`rimmer` ユーザーのログイン名を示します。

`500` ユーザー ID (UID) を示します。

`staff` ユーザーの一次グループを示します。

`10` グループ ID (GID) を示します。

`Annalee J. Rimmer` コメントを示します。

`/export/home/rimmer` ユーザーのホームディレクトリを示します。

`/bin/sh` ログインシェルを示します。

```
PS 010170 10 7 -1
```

次のパスワード有効期限情報を示します。

- パスワードの最終変更日
- 次に変更するまでに必要な日数
- 変更しないで使用できる日数
- 警告期間

▼ パスワードを持たないユーザーを表示する方法

1. **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

   Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2. **logins** コマンドを使用して、パスワードを持っていないユーザーをすべて表示します。

   ```
   # logins -p
   ```

   `-p` オプションを指定すると、パスワードを持たないユーザーが一覧表示されます。`logins` コマンドは、ネームサービスが有効になっていない限り、ローカルシステムのパスワードデータベースを使用します。

例 3–2 パスワードを持たないユーザーを表示する

次の例では、ユーザー `pmorph` はパスワードを持っていません。

```
# logins -p
pmorph          501     other           1       Polly Morph
#
```

▼ ユーザーのログインを一時的に無効にする方法

システムシャットダウンや定常的な保守の際にユーザーのログインを一時的に無効にします。スーパーユーザーのログインは影響を受けません。詳細は、`nologin(4)` のマニュアルページを参照してください。

1. **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

   Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 テキストエディタで、**/etc/nologin** ファイルを作成します。

```
# vi /etc/nologin
```

3 システムの利用に関するメッセージを入力します。

4 ファイルを閉じて、保存します。

例 3–3 ユーザーログインを無効にする

この例では、システムを使用できないことがユーザーに通知されます。

```
# vi /etc/nologin
(Add system message here)

# cat /etc/nologin
***No logins permitted.***

***The system will be unavailable until 12 noon.***
```

システムを実行レベル 0、つまりシングルユーザーモードにしてログインを無効にすることもできます。システムをシングルユーザーモードにする方法については、『Solaris のシステム管理 (基本編)』の第 10 章「システムのシャットダウン (手順)」を参照してください。

▼ ログイン失敗操作を監視する方法

この作業は、端末ウィンドウで行われたログイン操作の失敗を記録します。CDE ログインと ログインの操作失敗は記録しません。

1 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 **/var/adm** ディレクトリに **loginlog** ファイルを作成します。

```
# touch /var/adm/loginlog
```

3 **loginlog** ファイルに **root** ユーザーの読み取り権と書き込み権を設定します。

```
# chmod 600 /var/adm/loginlog
```

4 **loginlog** ファイルのグループのメンバーシップを **sys** に変更します。

```
# chgrp sys /var/adm/loginlog
```

5 ログが正常に記録されるか検証します。

たとえば、間違ったパスワードを使用してシステムに 5 回ログインします。次に、/var/adm/loginlog ファイルを表示します。

```
# more /var/adm/loginlog
jdoe:/dev/pts/2:Tue Nov  4 10:21:10 2010
jdoe:/dev/pts/2:Tue Nov  4 10:21:21 2010
jdoe:/dev/pts/2:Tue Nov  4 10:21:30 2010
jdoe:/dev/pts/2:Tue Nov  4 10:21:40 2010
jdoe:/dev/pts/2:Tue Nov  4 10:21:49 2010
#
```

loginlog ファイルには、失敗操作ごとに 1 つずつエントリが入っています。各エントリには、ユーザーのログイン名、tty デバイス、操作の失敗回数が入っています。4 回以下の失敗であれば、ログに記録されません。

loginlog ファイルのサイズが大きくなる場合は、コンピュータシステムへの侵入が試みられている可能性があります。このため、このファイルの内容を定期的にチェックして空にしてください。詳細は、loginlog(4) のマニュアルページを参照してください。

▼ すべてのログイン失敗操作を監視する方法

この作業は、失敗したログイン操作をすべて syslog ファイルに記録します。

1 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 **SYSLOG** と **SYSLOG_FAILED_LOGINS** に希望する値を指定して、**/etc/default/login** ファイルを設定します。

/etc/default/login ファイルを編集して、エントリを変更します。**SYSLOG=YES** のコメントを解除していることを確認してください。

```
# grep SYSLOG /etc/default/login
# SYSLOG determines whether the syslog(3) LOG_AUTH facility
# should be used
SYSLOG=YES
...
SYSLOG_FAILED_LOGINS=0
#
```

3 ログ操作の情報を記録できる適切なアクセス権でファイルを作成します。

a. **/var/adm** ディレクトリに **authlog** ファイルを作成します。

```
# touch /var/adm/authlog
```

b. **authlog** ファイルに、**root** ユーザーの読み取り権と書き込み権を設定します。

```
# chmod 600 /var/adm/authlog
```

c. **authlog** ファイルのグループのメンバーシップを **sys** に変更します。

```
# chgrp sys /var/adm/authlog
```

4 失敗したパスワード操作を記録するように、**syslog.conf** ファイルを編集します。
失敗は、authlog ファイルに送る必要があります。

a. **syslog.conf** ファイルに次のエントリを入力します。
syslog.conf 内の同じ行にあるフィールドは、タブで区切ります。

```
auth.notice      <Press Tab>   /var/adm/authlog
```

b. **syslog** デーモンの構成情報を再表示します。

```
# svcadm refresh system/system-log
```

5 ログが正常に記録されるか検証します。
たとえば、間違ったパスワードを使用し、通常のユーザーとしてシステムにログインします。続いて、Primary Administrator 役割かスーパーユーザーとして、/var/adm/authlog ファイルを表示します。

```
# more /var/adm/authlog
Nov  4 14:46:11 example1 login: [ID 143248 auth.notice]
 Login failure on /dev/pts/8 from example2, stacey
#
```

6 定期的に **/var/adm/authlog** ファイルを監視します。

例 3–4 ログインが 3 回失敗したあとのアクセス操作を記録する

/etc/default/login ファイルの SYSLOG_FAILED_LOGINS の値を 3 に設定する点以外は、前述の作業と同じです。

例 3–5 ログイン操作が 3 回失敗したあとで接続を遮断する

/etc/default/login ファイルの RETRIES エントリのコメントを解除し、RETRIES の値を 3 に設定します。この編集結果はただちに適用されます。1 つのセッションでログインが 3 回試されたあと、システムは接続を遮断します。

▼ ダイヤルアップパスワードを作成する方法

注意 – 最初にダイヤルアップパスワードを設定するときには、少なくとも 1 つのポートにログインしたまま別のポートでパスワードをテストしてください。ログアウトして新しいパスワードをテストすると、元どおりログインできなくなることがあります。まだ別のポートにログインしていれば、元に戻ってミスを訂正できます。

1. **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

   Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2. シリアルデバイスの一覧が入った **/etc/dialups** ファイルを作成します。

   このファイルには、ダイヤルアップパスワードで保護されているすべてのポートを含めてください。`/etc/dialups` ファイルは次のようになります。

   ```
   /dev/term/a
   /dev/term/b
   /dev/term/c
   ```

3. ダイヤルアップパスワードを要求するログインプログラムが入った **/etc/d_passwd** ファイルを作成します。

   `uucico`、 `sh`、 `ksh`、 `csh` など、ユーザーがログイン時に実行できるシェルプログラムを含めます。`/etc/d_passwd` ファイルは次のようになります。

   ```
   /usr/lib/uucp/uucico:encrypted-password:
   /usr/bin/csh:encrypted-password:
   /usr/bin/ksh:encrypted-password:
   /usr/bin/sh:encrypted-password:
   /usr/bin/bash:encrypted-password:
   ```

   この手順の後半で、各ログインプログラムに暗号化されたパスワードを追加することになります。

4. 2 つのファイルの所有権を **root** に設定します。

   ```
   # chown root /etc/dialups /etc/d_passwd
   ```

5. 2 つのファイルのグループの所有権を **root** に設定します。

   ```
   # chgrp root /etc/dialups /etc/d_passwd
   ```

6. 2 つのファイルに **root** の読み取り権と書き込み権を設定します。

   ```
   # chmod 600 /etc/dialups /etc/d_passwd
   ```

7 暗号化パスワードを作成します。

a. 一時的なユーザーを作成します。

```
# useradd username
```

b. 一時的なユーザーのパスワードを作成します。

```
# passwd username
New Password:       <Type password>
Re-enter new Password:        <Retype password>
passwd: password successfully changed for username
```

c. 暗号化パスワードを取り出します。

```
# grep username /etc/shadow > username.temp
```

d. *username*.**temp** ファイルを編集します。

暗号化パスワードを除くすべてのフィールドを削除します。2 つ目のフィールドに、暗号化パスワードが入っています。

たとえば、次の行では、暗号化パスワードは `U9gp9SyA/JlSk` です。

```
temp:U9gp9SyA/JlSk:7967:::::7988:
```

e. 一時的なユーザーを削除します。

```
# userdel username
```

8 *username*.**temp** ファイルから **/etc/d_passwd** ファイルに暗号化パスワードをコピーします。

ログインシェルごとに別のパスワードを作成することも、共通のパスワードを使用することもできます。

9 パスワードをダイヤルアップユーザーに知らせます。

盗聴のおそれがない方法でパスワードを知らせる必要があります。

▼ ダイヤルアップログインを一時的に無効にする方法

1 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 **/etc/d_passwd** ファイルのエントリを次の 1 行だけにします。

```
/usr/bin/sh:*:
```

パスワードアルゴリズムの変更 (作業マップ)

次の作業マップは、パスワードアルゴリズムを管理する手順を示しています。

作業	参照先
パスワードの強力な暗号化を可能にします	74 ページの「パスワード暗号化のアルゴリズムを指定する方法」
ネームサービスでパスワードの強力な暗号化を提供します	75 ページの「NIS ドメイン用の新しいパスワードアルゴリズムを指定する方法」
	76 ページの「NIS+ ドメイン用の新しいパスワードアルゴリズムを指定する方法」
	76 ページの「LDAP ドメイン用の新しいパスワードアルゴリズムを指定する方法」
新しいパスワード暗号化モジュールを追加します	77 ページの「Sun 以外のパスワード暗号化モジュールをインストールする方法」

パスワード暗号化のデフォルトアルゴリズムを変更する

デフォルトでは、ユーザーパスワードは `crypt_unix` アルゴリズムで暗号化されます。デフォルトのパスワード暗号化アルゴリズムを変更することによって、MD5 や Blowfish など、より強力な暗号化アルゴリズムを使用できます。

▼ パスワード暗号化のアルゴリズムを指定する方法

この手順では、ユーザーがパスワードを変更するときのデフォルト暗号化アルゴリズムとして BSD-Linux バージョンの MD5 アルゴリズムが使用されます。このアルゴリズムは、Oracle Solaris、 BSD、 Linux バージョンの UNIX が混在するマシンネットワークに適しています。パスワード暗号化アルゴリズムとアルゴリズム識別子の一覧は、表 2–1 を参照してください。

1 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 選択した暗号化アルゴリズムの識別子を指定します。

暗号化アルゴリズムの識別子を、/etc/security/policy.conf ファイルの CRYPT_DEFAULT 変数の値として入力します。

選択についての説明をコメントとして入力できます。

```
# cat  /etc/security/policy.conf
...
CRYPT_ALGORITHMS_ALLOW=1,2a,md5,5,6
#
# Use the version of MD5 that works with Linux and BSD systems.
# Passwords previously encrypted with __unix__ will be encrypted with MD5
# when users change their passwords.
#
#
CRYPT_DEFAULT=__unix__
CRYPT_DEFAULT=1
```

この例では、アルゴリズム構成を指定することによって、もっとも弱いアルゴリズムである crypt_unix がパスワードの暗号化に使用されることがないようにします。crypt_unix モジュールで暗号化されているパスワードは、次回のパスワード変更から crypt_bsdmd5 で暗号化されます。

選択したアルゴリズムの構成の詳細については、policy.conf(4) のマニュアルページを参照してください。

例 3–6 パスワードの暗号化に Blowfish アルゴリズムを使用する

この例では、policy.conf ファイルの CRYPT_DEFAULT 変数の値として、Blowfish アルゴリズムの識別子 2a が指定されています。

```
CRYPT_ALGORITHMS_ALLOW=1,2a,md5,5,6
#CRYPT_ALGORITHMS_DEPRECATE=__unix__
CRYPT_DEFAULT=2a
```

この構成は、Blowfish アルゴリズムを使用する BSD システムに対応しています。

▼ NIS ドメイン用の新しいパスワードアルゴリズムを指定する方法

NIS ドメインのユーザーがパスワードを変更すると、NIS クライアントは、/etc/security/policy.conf ファイルにある自身のローカルアルゴリズム構成を調べ、パスワードを暗号化します。

1 パスワード暗号化アルゴリズムを NIS クライアント上の /etc/security/policy.conf ファイルに指定します。

2 変更された /etc/security/policy.conf ファイルを NIS ドメインのすべてのクライアントマシンにコピーします。

3 混乱をできるだけ少なくするために、変更された **/etc/security/policy.conf** ファイルを **NIS** ルートサーバーとスレーブサーバーにコピーします。

▼ NIS+ ドメイン用の新しいパスワードアルゴリズムを指定する方法

NIS+ ドメインのユーザーがパスワードを変更すると、NIS+ ネームサービスは、NIS+ マスターにある /etc/security/policy.conf ファイルのアルゴリズム構成を調べます。rpc.nispasswd デーモンが動作するこの NIS+ マスターが、暗号化されたパスワードを作成します。

1 パスワード暗号化アルゴリズムを **NIS+** マスター上の **/etc/security/policy.conf** ファイルに指定します。

2 混乱をできるだけ少なくするために、**NIS+** マスターの **/etc/security/policy.conf** ファイルを **NIS+** ドメインのすべてのホストにコピーします。

▼ LDAP ドメイン用の新しいパスワードアルゴリズムを指定する方法

適切に構成された LDAP クライアントでは、新しいパスワードアルゴリズムを使用できます。LDAP クライアントは NIS クライアントと同じように動作します。

1 パスワード暗号化アルゴリズムを **LDAP** クライアント上の **/etc/security/policy.conf** ファイルに指定します。

2 変更された **policy.conf** ファイルを **LDAP** ドメインのすべてのクライアントマシンにコピーします。

3 クライアントの **/etc/pam.conf** ファイルが **pam_ldap** モジュールを使用していないことを確認します。

pam_ldap.so.1 を含むエントリの前にコメント記号 (#) があることを確認します。pam_authtok_store.so.1 モジュールには新しい server_policy オプションを使用しないでください。

ローカルアルゴリズム構成に基づくパスワードの暗号化は、クライアントの pam.conf ファイルの PAM エントリに従って行われます。また、パスワードの認証もこの PAM エントリによって行われます。

LDAP ドメインのユーザーがパスワードを変更すると、LDAP クライアントは、/etc/security/policy.conf ファイルにある自身のローカルアルゴリズム構成を調べ、パスワードを暗号化します。続いてクライアントは、{crypt} タグ付きの暗号

化パスワードをサーバーに送信します。このタグは、パスワードが暗号化済みであることをサーバーに知らせます。パスワードはそのままの形でサーバーに格納されます。認証時に、クライアントはこのパスワードをサーバーから取り出します。クライアントは、このパスワードと、入力されたユーザーのパスワードからクライアントが暗号化したばかりのパスワードとを比較します。

注 – LDAP サーバーでパスワードポリシー制御を使用するには、`pam.conf` ファイルの `pam_authtok_store` エントリに `server_policy` オプションを指定します。これによって、パスワードは、Sun Java System Directory Server の暗号化メカニズムを使ってサーバー上で暗号化されます。この手順については、『Solaris のシステム管理 (ネーミングとディレクトリサービス : DNS、NIS、LDAP 編)』の第 11 章「LDAP クライアントと Sun Java System Directory Server の設定 (手順)」を参照してください。

▼ Sun 以外のパスワード暗号化モジュールをインストールする方法

Sun 以外のパスワード暗号化アルゴリズムは、通常、ソフトウェアパッケージのモジュールの 1 つとして配布されます。したがって、`pkgadd` コマンドを実行すると、`/etc/security/crypt.conf` ファイルはベンダーのスクリプトによって変更されるはずです。このあとで、`/etc/security/policy.conf` ファイルに新しいモジュールとその識別子を指定してください。

1 **`pkgadd`** コマンドを実行してソフトウェアを追加します。

ソフトウェアの追加方法については、『Solaris のシステム管理 (基本編)』の「ソフトウェアパッケージの追加または削除 (pkgadd)」を参照してください。

2 新しいモジュールとモジュール識別子が追加されたことを確認します。

`/etc/security/crypt.conf` ファイル内の暗号化アルゴリズムの一覧でチェックしてください。

たとえば、次の行は、`crypt_rot13` アルゴリズムを実装するモジュールが追加されていることを示します。

```
# crypt.conf
#
md5 /usr/lib/security/$ISA/crypt_md5.so
rot13 /usr/lib/security/$ISA/crypt_rot13.so

# For *BSD - Linux compatibility
# 1 is MD5,  2a is Blowfish
1 /usr/lib/security/$ISA/crypt_bsdmd5.so
2a /usr/lib/security/$ISA/crypt_bsdbf.so
```

3 **/etc/security/policy.conf** ファイルに、新たにインストールされたアルゴリズムの識別子を追加します。

次に、識別子 rot13 を追加する必要がある policy.conf ファイルの一部を示します。

```
# Copyright 1999-2002 Sun Microsystems, Inc.  All rights reserved.
# ...
#ident  "@(#)policy.conf        1.12     08/05/14 SMI"
# ...
# crypt(3c) Algorithms Configuration
CRYPT_ALGORITHMS_ALLOW=1,2a,md5,5,6,,rot13
#CRYPT_ALGORITHMS_DEPRECATE=__unix__
CRYPT_DEFAULT=md5
```

この例では、現在のパスワードが crypt_rot13 アルゴリズムで暗号化されていれば、rot13 アルゴリズムが使用されます。新しいユーザーパスワードは crypt_sunmd5 アルゴリズムで暗号化されます。このアルゴリズム構成は Solaris だけからなるネットワークで有効です。

スーパーユーザーの監視と制限 (作業マップ)

次の作業マップは、root ユーザーログインの監視と制限の方法を示しています。

作業	説明	参照先
su コマンドを使用しているユーザーを監視します	sulog ファイルを定期的に走査します。	78 ページの「su コマンドを使用するユーザーを監視する方法」
スーパーユーザーの活動をコンソールに表示します	スーパーユーザーによるアクセス操作が行われたときそのアクセス操作を監視します。	79 ページの「スーパーユーザーのログインを制限し監視する方法」

スーパーユーザーの監視と制限

スーパーユーザーのアカウントを使用する代わりに、役割によるアクセス制御を設定できます。役割によるアクセス制御を RBAC と呼びます。RBAC の概要は、188 ページの「役割によるアクセス制御 (概要)」を参照してください。RBAC の設定方法については、第 9 章「役割によるアクセス制御の使用 (手順)」を参照してください。

▼ su コマンドを使用するユーザーを監視する方法

sulog ファイルには、ユーザーからスーパーユーザーに切り替えたときの su コマンドの使用を含め、すべての su コマンドの使用歴が記録されます。

- **/var/adm/sulog** ファイルの内容を定期的に監視します。

```
# more /var/adm/sulog
SU 12/20 16:26 + pts/0 stacey-root
```

```
SU 12/21 10:59 + pts/0 stacey-root
SU 01/12 11:11 + pts/0 root-rimmer
SU 01/12 14:56 + pts/0 pmorph-root
SU 01/12 14:57 + pts/0 pmorph-root
```

ここには、次のような情報が表示されます。

- コマンドが入力された日時。
- コマンド試行の成否。プラス記号 (+) は成功を示し、マイナス記号 (-) は失敗を示します。
- コマンドが実行されたポート。
- ユーザー名と切り替えたユーザー ID。

このファイルへの su ログの記録は、デフォルトで、 /etc/default/su ファイルの次のエントリで有効になっています。

```
SULOG=/var/adm/sulog
```

注意事項 ??? を含むエントリは、su コマンドの制御端末を識別できないことを示しています。通常、デスクトップが表示される前の su コマンドのシステム呼び出しには、??? が含まれます。たとえば、SU 10/10 08:08 + ??? root-root です。ユーザーがデスクトップセッションを開始すると、ttynam コマンドは、次のように制御端末の値を sulog に返します。 SU 10/10 10:10 + pts/3 jdoe-root。

次のようなエントリは、su コマンドがコマンド行で呼び出されなかったことを示している場合があります。SU 10/10 10:20 + ??? root-oracle。ユーザーが GUI を使用して oracle ロールに切り替えた可能性があります。

▼ スーパーユーザーのログインを制限し監視する方法

この方法では、ローカルシステムにアクセスしようとするスーパーユーザーをただちに検出できます。

1 **/etc/default/login ファイルの CONSOLE エントリを確認します。**

```
CONSOLE=/dev/console
```

デフォルトのコンソールデバイスは /dev/console に設定されています。このように設定されていると、root はコンソールにログインできます。root は遠隔ログインを行うことはできません。

2 **root** が遠隔ログインできないことを検証します。

遠隔システムから、スーパーユーザーとしてログインを試みます。

```
mach2 % rlogin -l root mach1
Password:      <Type root password of mach1>
Not on system console
Connection closed.
```

3 スーパーユーザーになろうとする試みを監視します。

デフォルトでは、スーパーユーザーになろうとすると、その試行が SYSLOG ユーティリティーによってコンソールに表示されます。

a. デスクトップに端末コンソールを開きます。

b. 別のウインドウで、**su** コマンドを使ってスーパーユーザーになります。

```
% su -
Password:      <Type root password>
#
```

端末コンソールにメッセージが表示されます。

```
Sep 7 13:22:57 mach1 su: 'su root' succeeded for jdoe on /dev/pts/6
```

例 3–7 スーパーユーザーのアクセス試行のログを作成する

この例では、スーパーユーザーになろうとする試みは SYSLOG によってログされていません。そのため、管理者は、/etc/default/su ファイルの #CONSOLE=/dev/console エントリのコメントを解除して、試行のログを作成します。

```
# CONSOLE determines whether attempts to su to root should be logged
# to the named device
#
CONSOLE=/dev/console
```

ユーザーがスーパーユーザーになろうとすると、その試行が端末コンソールに表示されます。

```
SU 09/07 16:38 + pts/8 jdoe-root
```

注意事項 /etc/default/login ファイルにデフォルトの CONSOLE エントリが含まれている場合、遠隔システムからスーパーユーザーになるには、ユーザーはまず自分のユーザー名でログインする必要があります。自分のユーザー名でログインしたあとに、su コマンドを使ってスーパーユーザーになることができます。

コンソールに Mar 16 16:20:36 mach1 login: ROOT LOGIN /dev/pts/14 FROM mach2.Example.COM のようなエントリが表示されたら、システムは遠隔 root ログイン

を認めていることになります。遠隔システムからのスーパーユーザーアクセスを禁止するには、/etc/default/login ファイルの #CONSOLE=/dev/console エントリを CONSOLE=/dev/console に変更します。

SPARC: システムハードウェアに対するアクセスの制御 (作業マップ)

次の作業マップは、PROM を不当なアクセスから守る方法を示しています。

作業	説明	参照先
ユーザーにシステムのハードウェア設定を変更させません	PROM 設定の変更にパスワードを求めます。	81 ページの「ハードウェアアクセスのパスワードを必須にする方法」
アボートシーケンスを無効にします	ユーザーが PROM にアクセスできないようにします。	82 ページの「システムのアボートシーケンスを無効にする方法」

システムハードウェアアクセスの制御

物理的なマシンは、ハードウェア設定にアクセスする際にパスワードを入力させることで保護できます。さらに、ユーザーがアボートシーケンスを使ってウィンドウシステムを離れるのを防ぐことによってマシンを保護する方法もあります。

▼ ハードウェアアクセスのパスワードを必須にする方法

x86 システムでは、BIOS の保護が PROM の保護に相当します。BIOS を保護する方法については、使用しているマシンのマニュアルを参照してください。

1. スーパーユーザーになるか、あるいは **Device Security** プロファイル、**Maintenance and Repair** プロファイル、または **System Administrator** プロファイルを含んだ役割を引き受けます。

   System Administrator プロファイルには、Maintenance and Repair プロファイルが含まれます。System Administrator プロファイルを含む役割の作成や、役割をユーザーに割り当てる方法については、210 ページの「RBAC の構成 (作業マップ)」を参照してください。

2. 端末ウィンドウで、次のように入力して **PROM** セキュリティーモードに入ります。

   ```
   # eeprom security-mode=command
   ```

```
Changing PROM password:
New password:      <Type password>
Retype new password:      <Retype password>
```

値として command か full を選択します。詳細は、eeprom(1M) のマニュアルページを参照してください。

前述のコマンドを入力する際に PROM パスワードを要求されない場合は、システムがすでに PROM パスワードを持っています。

3 (省略可能) **PROM** パスワードを変更する場合は、次のコマンドを入力します。

```
# eeprom security-password=      Press Return
Changing PROM password:
New password:      <Type password>
Retype new password:      <Retype password>
```

新しい PROM セキュリティーモードとパスワードはただちに有効になりますが、それが認識できるのは、ほとんどの場合、次回の起動時です。

注意 – PROM パスワードを忘れないでください。このパスワードがないと、ハードウェアは使用できません。

▼ システムのアボートシーケンスを無効にする方法

一部のサーバーシステムにはキースイッチがあります。このキースイッチを安全な位置に設定すると、ソフトウェアキーボードのアボート設定が無効になります。そのため、次の手順で行った変更が実装されないことがあります。

1 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 **KEYBOARD_ABORT** の値を **disable** に変更します。

/etc/default/kbd ファイルの enable 行をコメントにします。次に disable 行を追加します。

```
# cat /etc/default/kbd
...
# KEYBOARD_ABORT affects the default behavior of the keyboard abort
# sequence, see kbd(1) for details.  The default value is "enable".
# The optional value is "disable".  Any other value is ignored.
...
#KEYBOARD_ABORT=enable
KEYBOARD_ABORT=disable
```

3 キーボードのデフォルトを更新します。

```
# kbd -i
```

◆ ◆ ◆ 第 4 章

4 デバイスアクセスの制御 (作業)

この章では、デバイスを保護するための作業について説明するとともに、参考となる節を示します。この章の内容は次のとおりです。

デバイスの保護についての概要は、49 ページの「デバイスアクセスの制御」を参照してください。

デバイスの構成 (作業マップ)

次の作業マップは、デバイスへのアクセスを管理する作業を示しています。

作業	参照先
デバイスポリシーを管理します	86 ページの「デバイスポリシーの設定 (作業マップ)」
デバイス割り当てを管理します	89 ページの「デバイス割り当ての管理 (作業マップ)」
デバイス割り当てを実行します	95 ページの「デバイスの割り当て (作業マップ)」

デバイスポリシーの設定 (作業マップ)

次の作業マップは、デバイスポリシーに関連するデバイス構成作業の参照先を示しています。

作業	説明	参照先
システム上のデバイスのデバイスポリシーを確認します	デバイスとそれらのデバイスポリシーの一覧を表示します。	86 ページの「デバイスポリシーを表示する方法」
デバイス使用に対して特権を要求します	特権を使用してデバイスを保護します。	87 ページの「既存のデバイスのデバイスポリシーを変更する方法」
デバイスから特権要件を削除します	デバイスのアクセスに必要な特権を削除するか、そのレベルを下げます。	例 4–3
デバイスポリシーの変更を監査します	監査トレールでデバイスポリシーの変更を記録します	88 ページの「デバイスポリシーの変更を監査する方法」
`/dev/arp` にアクセスします	Oracle Solaris IP MIB-II 情報を取得します。	88 ページの「`/dev/*` デバイスから IP MIB-II 情報を取得する方法」

デバイスポリシーの設定

デバイスポリシーは、システムに不可欠なデバイスに対するアクセスの制限または防止を行うものです。このポリシーはカーネルで適用されます。

▼ デバイスポリシーを表示する方法

- システム上のすべてのデバイスのデバイスポリシーを表示します。

```
% getdevpolicy | more
DEFAULT
        read_priv_set=none
        write_priv_set=none
ip:*
        read_priv_set=net_rawaccess
        write_priv_set=net_rawaccess
...
```

例 4–1 特定のデバイスのデバイスポリシーを表示する

この例では、3 つのデバイスのデバイスポリシーが表示されています。

```
% getdevpolicy /dev/allkmem /dev/ipsecesp /dev/hme
/dev/allkmem
        read_priv_set=all
```

```
        write_priv_set=all
/dev/ipsecesp
        read_priv_set=sys_net_config
        write_priv_set=sys_net_config
/dev/hme
        read_priv_set=net_rawaccess
        write_priv_set=net_rawaccess
```

▼ 既存のデバイスのデバイスポリシーを変更する方法

1 **Device Security 権利プロファイルを含む役割を引き受けるか、あるいはスーパーユーザーになります。**

Primary Administrator 役割には、Device Security 権利プロファイルが含まれます。また、作成する役割に Device Security 権利プロファイルを割り当てることもできます。役割を作成してその役割をユーザーに割り当てる方法については、例 9–3 を参照してください。

2 デバイスにポリシーを追加します。

```
# update_drv -a -p policy device-driver
```

`-a`	*device-driver* 用の *policy* を指定します。
`-p` *policy*	*device-driver* のデバイスポリシーです。デバイスポリシーは、2 セットの特権を指定します。1 つは、デバイスの読み取りに必要です。もう 1 つは、デバイスへの書き込みに必要です。
device-driver	デバイスドライバです。

詳細は、`update_drv(1M)` のマニュアルページを参照してください。

例 4–2 既存のデバイスにポリシーを追加する

次の例では、デバイス `ipnat` にデバイスポリシーが追加されています。

```
# getdevpolicy /dev/ipnat
/dev/ipnat
        read_priv_set=none
        write_priv_set=none
# update_drv -a \
-p 'read_priv_set=net_rawaccess write_priv_set=net_rawaccess' ipnat
# getdevpolicy /dev/ipnat
/dev/ipnat
        read_priv_set=net_rawaccess
        write_priv_set=net_rawaccess
```

例 4–3 デバイスからポリシーを削除する

次の例では、デバイス ipnat のデバイスポリシーから読み取り特権セットが削除されます。

```
# getdevpolicy /dev/ipnat
/dev/ipnat
        read_priv_set=net_rawaccess
        write_priv_set=net_rawaccess
# update_drv -a -p write_priv_set=net_rawaccess ipnat
# getdevpolicy /dev/ipnat
/dev/ipnat
        read_priv_set=none
        write_priv_set=net_rawaccess
```

▼ デバイスポリシーの変更を監査する方法

デフォルトでは、as 監査クラスに、AUE_MODDEVPLCY 監査イベントが含まれます。

1 **Primary Administrator 役割を引き受けるか、スーパーユーザーになります。**

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 **AUE_MODDEVPLCY 監査イベントを含む監査クラスをあらかじめ選択します。**

audit_control ファイルの flags 行に as クラスを追加してください。このファイルは次のようになります。

```
# audit_control file
dir:/var/audit
flags:lo,as
minfree:20
naflags:lo
```

詳しい操作説明は、625 ページの「audit_control ファイルの変更方法」を参照してください。

▼ /dev/* デバイスから IP MIB-II 情報を取得する方法

Oracle Solaris IP MIB-II 情報を取得するアプリケーションは、/dev/ip ではなく /dev/arp を開く必要があります。

1 **/dev/ip および /dev/arp のデバイスポリシーを決定します。**

```
% getdevpolicy /dev/ip /dev/arp
/dev/ip
        read_priv_set=net_rawaccess
```

```
        write_priv_set=net_rawaccess
/dev/arp
        read_priv_set=none
        write_priv_set=none
```

/dev/ip の読み取りおよび書き込みには、net_rawaccess 特権が必要であることに注意してください。/dev/arp は特権を必要としません。

2 **/dev/arp** を開き、**tcp** モジュールと **udp** モジュールをプッシュします。

特権は不要です。この方法は、/dev/ip を開いて arp、 tcp、 および udp モジュールをプッシュするのと同じです。現在、/dev/ip を開くには特権が必要なため、/dev/arp メソッドを推奨します。

デバイス割り当ての管理 (作業マップ)

次の作業マップは、デバイス割り当ての有効化と設定について説明した箇所を示しています。デフォルトではデバイス割り当ては有効になっていません。デバイス割り当てを有効にしたあとで、95 ページの「デバイスの割り当て (作業マップ)」を参照してください。

作業	説明	参照先
デバイスを割り当て可能にします	デバイスを一度に 1 人のユーザーに割り当てられるようにします。	90 ページの「デバイスを割り当て可能にする方法」
ユーザーによるデバイス割り当てを承認します	デバイス割り当ての承認をユーザーに与えます。	90 ページの「ユーザーによるデバイス割り当てを承認する方法」
システム上の割り当て可能なデバイスを表示します	割り当てが可能なデバイスと、そのデバイスの状態を一覧表示します。	91 ページの「デバイスの割り当て情報を表示する方法」
デバイスを強制的に割り当てます	ただちにデバイスを使う必要があるユーザーにデバイスを割り当てます	92 ページの「デバイスの強制的な割り当て」
デバイスの割り当てを強制的に解除します	現在ユーザーに割り当てられているデバイスの割り当てを解除します	93 ページの「デバイスの強制的な割り当て解除」
デバイスの割り当てプロパティーを変更します	デバイス割り当ての要件を変更します	93 ページの「割り当て可能デバイスの変更方法」
デバイスクリーンスクリプトを作成します	物理デバイスからデータを一掃します。	107 ページの「新しいデバイスクリーンスクリプトの作成」
デバイス割り当てを無効にします	すべてのデバイスの割り当て制限を解除します。	644 ページの「監査サービスを無効にする方法」
デバイス割り当ての監査を行います	デバイス割り当てを監査トレールに記録します	94 ページの「デバイス割り当てを監査する方法」

デバイス割り当ての管理

デバイス割り当ては、周辺機器に対するアクセスの制限または防止を行う作業です。制限は、ユーザーの割り当て時に適用されます。デフォルトでは、割り当て可能デバイスにアクセスする場合、ユーザーは承認を必要とします。

▼ デバイスを割り当て可能にする方法

すでに bsmconv コマンドを実行して監査を有効にしている場合、システムではすでにデバイス割り当てが有効になっています。詳細は、bsmconv(1M) のマニュアルページを参照してください。

1 **Audit Control 権利プロファイルを含む役割を引き受けるか、あるいはスーパーユーザーになります。**

Primary Administrator 役割には、Audit Control 権利プロファイルが含まれます。また、作成する役割に Audit Control 権利プロファイルを割り当てることもできます。役割を作成してその役割をユーザーに割り当てる方法については、例 9–3 を参照してください。

2 デバイス割り当てを有効にします。

```
# bsmconv
This script is used to enable the Basic Security Module (BSM).
Shall we continue with the conversion now? [y/n] y
bsmconv: INFO: checking startup file.
bsmconv: INFO: move aside /etc/rc3.d/S81volmgt.
bsmconv: INFO: turning on audit module.
bsmconv: INFO: initializing device allocation files.

The Basic Security Module is ready.
If there were any errors, please fix them now.
Configure BSM by editing files located in /etc/security.
Reboot this system now to come up with BSM enabled.
```

注 – Volume Management デーモン (/etc/rc3.d/S81volmgt) は、このコマンドによって無効になります。

▼ ユーザーによるデバイス割り当てを承認する方法

1 **Primary Administrator 役割を引き受けるか、スーパーユーザーになります。**

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 適切な承認とコマンドが入った権利プロファイルを作成します。

一般には、solaris.device.allocate 承認を含む権利プロファイルを作成します。234 ページの「権利プロファイルを作成または変更する方法」に挙げられている説明に従って操作してください。権利プロファイルに、次に示すような適切なプロパティーを指定します。

- 権利プロファイル名: Device Allocation
- 付与される承認: solaris.device.allocate
- セキュリティー属性を指定したコマンド: sys_mount 特権を指定した mount、sys_mount 特権を指定した umount

3 権利プロファイルの役割を作成します。

213 ページの「GUI を使用して役割の作成および割り当てを行う方法」に挙げられている説明に従って操作してください。次に示す役割プロパティーを参考にしてください。

- 役割名: devicealloc
- 役割の完全名: Device Allocator
- 役割の説明: Allocates and mounts allocated devices
- 権利プロファイル: Device Allocation

  この権利プロファイルは、役割に含めるプロファイルのリストの先頭に配置する必要があります。

4 デバイス割り当てを許可する各ユーザーに、この役割を割り当てます。

5 これらのユーザーにデバイス割り当ての方法を教えます。

リムーバブルメディアの割り当て例は、95 ページの「デバイスを割り当てる方法」を参照してください。

Volume Management デーモン (vold) が稼働していないため、リムーバブルメディアは自動的にはマウントされません。割り当て済みのデバイスをマウントする例については、96 ページの「割り当て済みデバイスをマウントする方法」を参照してください。

▼ デバイスの割り当て情報を表示する方法

始める前に この作業を行うには、デバイス割り当てが有効になっていなければなりません。デバイス割り当てを有効にする方法については、90 ページの「デバイスを割り当て可能にする方法」を参照してください。

1. **Device Security 権利プロファイルを含む役割を引き受けるか、あるいはスーパーユーザーになります。**

   Primary Administrator 役割には、Device Security 権利プロファイルが含まれます。また、作成する役割に Device Security 権利プロファイルを割り当てることもできます。役割を作成してその役割をユーザーに割り当てる方法については、例 9–3 を参照してください。

2. **システム上の割り当て可能デバイスについての情報を表示します。**

   ```
   # list_devices device-name
   ```

   device-name は次のいずれかです。

   - `audio`[*n*] – マイクとスピーカーです。
   - `fd`[*n*] – フロッピーディスクドライブです。
   - `sr`[*n*] – CD-ROM ドライブです。
   - `st`[*n*] – テープドライブです。

注意事項 `list_devices` コマンドが次のようなエラーメッセージを返す場合は、デバイス割り当てが有効になっていないか、情報を取得するために必要なアクセス権がありません。

```
list_devices: No device maps file entry for specified device.
```

コマンドを実行するには、デバイス割り当てを有効にし、`solaris.device.revoke` 承認のある役割を引き受けてください。

▼ デバイスの強制的な割り当て

強制的な割り当ては、誰かがデバイスの割り当て解除を忘れた場合や、デバイスをただちに使用する必要がある場合などに行います。

始める前に この処理を行うには、ユーザーまたは役割に `solaris.device.revoke` 承認がなければなりません。

1. **自分の役割に適切な承認が含まれているか確認します。**

   ```
   $ auths
   solaris.device.allocate solaris.device.revoke
   ```

2. **デバイスを必要としているユーザーにデバイスを強制的に割り当てます。**

   この例では、テープドライブがユーザー `jdoe` に強制的に割り当てられます。

   ```
   $ allocate -U jdoe
   ```

▼ デバイスの強制的な割り当て解除

ユーザーが割り当てたデバイスは、プロセスの終了時やそのユーザーのログアウトの際に自動的に割り当てが解除されることはありません。強制的な割り当て解除は、ユーザーがデバイスの割り当てを解除することを忘れた場合に行います。

始める前に　この処理を行うには、ユーザーまたは役割に `solaris.device.revoke` 承認がなければなりません。

1. 自分の役割に適切な承認が含まれているか確認します。

```
$ auths
solaris.device.allocate solaris.device.revoke
```

2. デバイスの割り当てを強制的に解除します。

この例では、プリンタの割り当てが強制的に解除されます。現在、このプリンタはほかのユーザーが割り当てを行える状態にあります。

```
$ deallocate -f /dev/lp/printer-1
```

▼ 割り当て可能デバイスの変更方法

1. **Device Security** 権利プロファイルを含む役割を引き受けるか、あるいはスーパーユーザーになります。

Primary Administrator 役割には、Device Security 権利プロファイルが含まれます。また、作成する役割に Device Security 権利プロファイルを割り当てることもできます。役割を作成してその役割をユーザーに割り当てる方法については、例 9–3 を参照してください。

2. 承認が必要であるかどうかを指定するか、あるいは **`solaris.device.allocate`** 承認を指定します。

`device_allocate` ファイルのデバイスエントリにある 5 つ目のフィールドを変更します。

```
audio;audio;reserved;reserved;solaris.device.allocate;/etc/security/lib/audio_clean
fd0;fd;reserved;reserved;solaris.device.allocate;/etc/security/lib/fd_clean
sr0;sr;reserved;reserved;solaris.device.allocate;/etc/security/lib/sr_clean
```

`solaris.device.allocate` は、デバイスの使用に `solaris.device.allocate` 承認が必要であることを示します。

例 4–4　任意のユーザーによるデバイス割り当てを許可する

次の例では、システム上のどのユーザーも任意のデバイスを割り当てることができます。`device_allocate` ファイルの各デバイスエントリ内にある 5 番目のフィールドは、単価記号 (@) に変更されました。

```
$ whoami
devicesec
$ vi /etc/security/device_allocate
audio;audio;reserved;reserved;@;/etc/security/lib/audio_clean
fd0;fd;reserved;reserved;@;/etc/security/lib/fd_clean
sr0;sr;reserved;reserved;@;/etc/security/lib/sr_clean
...
```

例 4–5 一部の周辺機器の使用を防止する

次の例では、オーディオデバイスの使用が禁止されています。device_allocate ファイルのオーディオデバイスエントリにある 5 番目のフィールドは、アスタリスク (*) に変更されました。

```
$ whoami
devicesec
$ vi /etc/security/device_allocate
audio;audio;reserved;reserved;*;/etc/security/lib/audio_clean
fd0;fd;reserved;reserved;solaris device.allocate;/etc/security/lib/fd_clean
sr0;sr;reserved;reserved;solaris device.allocate;/etc/security/lib/sr_clean
...
```

例 4–6 すべての周辺機器の使用を防止する

次の例では、使用できる周辺機器はありません。device_allocate ファイルの各デバイスエントリにある 5 番目のフィールドは、アスタリスク (*) に変更されました。

```
$ whoami
devicesec
$ vi /etc/security/device_allocate
audio;audio;reserved;reserved;*;/etc/security/lib/audio_clean
fd0;fd;reserved;reserved;*;/etc/security/lib/fd_clean
sr0;sr;reserved;reserved;*;/etc/security/lib/sr_clean
...
```

▼ デバイス割り当てを監査する方法

デフォルトでは、デバイス割り当てコマンドは、監査クラス other の状態です。

1. **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

   Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 監査の対象となるように、**ot** クラスをあらかじめ選択します。

`audit_control` ファイルの `flags` 行にクラス `ot` を追加してください。このファイルは次のようになります。

```
# audit_control file
dir:/var/audit
flags:lo,ot
minfree:20
naflags:lo
```

詳しい操作説明は、625 ページの「`audit_control` ファイルの変更方法」を参照してください。

デバイスの割り当て (作業マップ)

次の作業マップは、デバイスの割り当て方法を説明した手順を示しています。

作業	説明	参照先
デバイスを割り当てます	デバイスの使用を 1 人のユーザーだけに限定し、ほかのユーザーがそのデバイスを使用できないようにします。	95 ページの「デバイスを割り当てる方法」
割り当てられたデバイスをマウントします	マウントが必要なデバイス (CD-ROM やフロッピーディスクなど) をユーザーが確認できるようにします。	96 ページの「割り当て済みデバイスをマウントする方法」
デバイスの割り当てを解除します	割り当て可能なデバイスをほかのユーザーが使用できるようにします。	99 ページの「デバイスの割り当てを解除する方法」

デバイスの割り当て

デバイス割り当ては、一度に 1 人のユーザーだけが使用できるようにデバイスを予約 (確保) する作業です。マウントポイントが必要なデバイスはマウントする必要があります。

▼ デバイスを割り当てる方法

始める前に デバイス割り当ての有効化は、90 ページの「デバイスを割り当て可能にする方法」に説明されている方法で行う必要があります。承認が必要な場合は、そのユーザーは承認を得ていなければなりません。

1 デバイスを割り当てます。

デバイス名でデバイスを指定します。

```
% allocate device-name
```

2 デバイスが割り当てられたことを検証します。

同じコマンドをもう一度実行します。

```
% allocate device-name
allocate. Device already allocated.
```

例 4–7 マイクを割り当てる

この例では、ユーザー jdoe がマイク audio を割り当てます。

```
% whoami
jdoe
% allocate audio
```

例 4–8 プリンタを割り当てる

この例では、ユーザーがプリンタを割り当てます。このユーザーが printer-1 の割り当てを解除するか、このプリンタが強制的にほかのユーザーに割り当てられるまで、ほかのユーザーはこのプリンタを使用できません。

```
% allocate /dev/lp/printer-1
```

強制的な割り当て解除の例は、93 ページの「デバイスの強制的な割り当て解除」を参照してください。

例 4–9 テープドライブを割り当てる

この例では、ユーザー jdoe がテープドライブ st0 を割り当てます。

```
% whoami
jdoe
% allocate st0
```

注意事項 allocate コマンドがデバイスを割り当てることができない場合は、コンソールウィンドウにエラーメッセージが表示されます。割り当てのエラーメッセージについては、allocate(1) のマニュアルページを参照してください。

▼ 割り当て済みデバイスをマウントする方法

始める前に ユーザーまたは役割によってすでにデバイスが割り当てられています。デバイスをマウントするには、そのデバイスのマウントに必要な特権をそのユーザーまたは役割が保持していなければなりません。必要な特権を与える方法については、90 ページの「ユーザーによるデバイス割り当てを承認する方法」を参照してください。

1. デバイスの割り当てまたはマウントが行える役割を引き受けます。

```
% su - role-name
Password:       <Type role-name password>
$
```

2. この役割のホームディレクトリにマウントポイントを作成し、このマウントポイントを保護します。

   このステップが必要なのは、マウントポイントが初めて必要になった時だけです。

```
$ mkdir mount-point ; chmod 700 mount-point
```

3. 割り当てが可能なデバイスを一覧表示します。

```
$ list_devices -l
List of allocatable devices
```

4. デバイスを割り当てます。

   デバイス名でデバイスを指定します。

```
$ allocate device-name
```

5. デバイスをマウントします。

```
$ mount -o ro -F filesystem-type device-path mount-point
```

   各情報の意味は次のとおりです。

`-o ro`	デバイスは読み取り専用としてマウントされることを示します。デバイスに書き込みができるように指定するには、`-o rw` を使用します。
`-F` *filesystem-type*	デバイスのファイルシステムフォーマットを示します。一般に、CD-ROM は HSFS ファイルシステムでフォーマットされています。フロッピーディスクは、一般に PCFS ファイルシステムでフォーマットされています。
device-path	デバイスへのパスを示します。`list_devices -l` コマンドの出力には、*device-path* が含まれます。
mount-point	手順 2 で作成したマウントポイントを示します。

例 4–10 フロッピーディスクドライブの割り当て

この例では、ユーザーはフロッピーディスクドライブ `fd0` の割り当てとマウントが行える役割を引き受けます。このフロッピーディスクは、PCFS ファイルシステムでフォーマットされています。

```
% roles
devicealloc
% su - devicealloc
Password:       <Type devicealloc password>
```

```
$ mkdir /home/devicealloc/mymnt
$ chmod 700 /home/devicealloc/mymnt
$ list_devices -l
...
device: fd0 type: fd files: /dev/diskette /dev/rdiskette /dev/fd0a
...
$ allocate fd0
$ mount -o ro -F pcfs /dev/diskette /home/devicealloc/mymnt
$ ls /home/devicealloc/mymnt
```

List of the contents of diskette

例 4–11 CD-ROM ドライブを割り当てる

この例では、ユーザーは CD-ROM ドライブ `sr0` の割り当てとマウントが行える役割を引き受けます。このドライブは、HSFS ファイルシステムでフォーマットされています。

```
% roles
devicealloc
% su - devicealloc
Password:       <Type devicealloc password>
$ mkdir /home/devicealloc/mymnt
$ chmod 700 /home/devicealloc/mymnt
$ list_devices -l
...
device: sr0 type: sr files: /dev/sr0 /dev/rsr0 /dev/dsk/c0t2d0s0 ...
...
$ allocate sr0
$ mount -o ro -F hsfs /dev/sr0 /home/devicealloc/mymnt
$ cd /home/devicealloc/mymnt ; ls
```

List of the contents of CD-ROM

注意事項 `mount` コマンドがデバイスをマウントできない場合は、「`mount: insufficient privileges`」というエラーメッセージが表示されます。次の点を確認します。

- `mount` コマンドをプロファイルシェルで実行していることを確認します。役割を引き受けた場合は、その役割にプロファイルシェルがあります。`mount` コマンドでプロファイルを割り当てられたユーザーの場合、プロファイルシェルを作成する必要があります。プロファイルシェルの作成には、コマンド `pfsh`、`pfksh`、および `pfcsh` を使用します。
- 指定されたマウントポイントを所有しているかを確認します。このマウントポイントに対して読み取り、書き込み、および実行のアクセス権がなければなりません。

以上の要件を満たしているにもかかわらず割り当て済みデバイスをマウントできないという場合は、管理者に問い合わせてください。

▼ デバイスの割り当てを解除する方法

割り当てを解除すると、ほかのユーザーもユーザーの使用後にそのデバイスを割り当てて使用できるようになります。

始める前に　デバイスをすでに割り当てていなければなりません。

1. デバイスがマウントされている場合は、デバイスのマウントを解除します。

```
$ cd $HOME
$ umount mount-point
```

2. デバイスの割り当てを解除します。

```
$ deallocate device-name
```

例 4–12　マイクの割り当てを解除する

この例では、ユーザー jdoe がマイク audio の割り当てを解除します。

```
% whoami
jdoe
% deallocate audio
```

例 4–13　CD-ROM ドライブの割り当てを解除する

この例では、Device Allocator 役割が、CD-ROM ドライブの割り当てを解除します。次のメッセージが表示されたあとで、CD-ROM が取り出されます。

```
$ whoami
devicealloc
$ cd /home/devicealloc
$ umount /home/devicealloc/mymnt
$ ls /home/devicealloc/mymnt
$
$ deallocate sr0
/dev/sr0:      326o
/dev/rsr0:     326o
...
sr_clean: Media in sr0 is ready.  Please, label and store safely.
```

デバイスの保護 (参照)

Oracle Solaris では、デバイスはデバイスポリシーによって保護されます。周辺機器は、デバイス割り当てによって保護できます。デバイスポリシーはカーネルによって適用されます。デバイス割り当ては、ユーザーレベルで任意に有効化と適用が行われます。

デバイスポリシーコマンド

デバイス管理コマンドは、ローカルファイルのデバイスポリシーを管理するコマンドです。デバイスポリシーは特権要件を含むことができます。デバイスを管理できるのは、スーパーユーザーと、スーパーユーザーと同等の能力を持つ役割だけです。

次の表は、デバイス管理コマンドを示しています。

表 4–1 デバイス管理コマンド

コマンド	目的	マニュアルページ
devfsadm	稼働しているシステム上のデバイスとデバイスドライバを管理します。また、デバイスポリシーの読み込みも行います。 devfsadm コマンドは、ディスクデバイス、テープデバイス、ポートデバイス、オーディオデバイス、および擬似デバイスに対する /dev リンクのクリーンアップにも使用できます。名前付きドライバのデバイスの再構成も行えます。	devfsadm(1M)
getdevpolicy	1 つ以上のデバイスに関連付けられたポリシーを表示します。このコマンドはどのユーザーでも実行できます。	getdevpolicy(1M)
add_drv	稼働中のシステムに新しいデバイスドライバを追加します。新しいデバイスにデバイスポリシーを追加するオプションを含みます。一般に、このコマンドはデバイスドライバのインストール中にスクリプト内で呼び出されます。	add_drv(1M)
update_drv	既存のデバイスドライバの属性を更新します。デバイスのデバイスポリシーを更新するオプションを含みます。一般に、このコマンドはデバイスドライバのインストール中にスクリプト内で呼び出されます。	update_drv(1M)
rem_drv	デバイスまたはデバイスドライバを削除します。	rem_drv(1M)

デバイスの割り当て

デバイス割り当てによって、データの消失、コンピュータウイルス、セキュリティー侵害などからサイトを保護できます。デバイスポリシーと違い、デバイス割り当ては任意です。デバイスは、bsmconv スクリプトが実行されるまで割り当てることはできません。デバイス割り当ては、割り当て可能デバイスへのアクセスを制限するのに承認を使用します。

デバイス割り当ての構成要素

デバイス割り当てメカニズムの構成要素は、次のとおりです。

- allocate、 deallocate、 dminfo、 list_devices コマンド。詳細は、101 ページの「デバイス割り当てコマンド」を参照してください。
- 各割り当て可能デバイスのデバイスクリーンスクリプト。

これらのコマンドとスクリプトは、次のローカルファイルを使用してデバイス割り当てを実装します。

- /etc/security/device_allocate ファイル。詳細は、device_allocate(4) のマニュアルページを参照してください。
- /etc/security/device_maps ファイル。詳細は、device_maps(4) のマニュアルページを参照してください。
- ロックファイル。割り当て可能デバイスごとに /etc/security/dev ディレクトリに配置します。
- 各割り当て可能デバイスに関連付けられたロックファイルの変更後の属性。

注 – /etc/security/dev ディレクトリは、将来の Oracle Solaris リリースでサポートされなくなる可能性があります。

デバイス割り当てコマンド

大文字のオプションが指定された allocate、 deallocate、 および list_devices コマンドは管理用コマンドです。それ以外ではこれらのコマンドはユーザーコマンドです。次の表は、デバイス割り当てコマンドを示しています。

表 4–2 デバイス割り当てコマンド

コマンド	目的	マニュアルページ
bsmconv	デバイス割り当てを処理するデータベースを作成します。監査サービスの有効化も行います。スーパーユーザーであるか、Primary Administrator 役割を引き受ける必要があります。 devalloc_adm - デバイスを割り当て可能にすることで、個々のユーザーはデバイスを個人使用目的で割り当てできます。デバイス割り当てを禁止することで、システムの周辺機器の使用を禁止します。割り当て可能デバイスの一覧からデバイスを削除します。 監査を使用しない場合、devalloc_adm コマンドを使用してデバイス割り当てを有効にすることができます。	bsmconv(1M)

表 4–2 デバイス割り当てコマンド (続き)

コマンド	目的	マニュアルページ
dminfo	デバイスタイプ、デバイス名、またはフルパス名を指定して、割り当て可能デバイスを検索します。	dminfo(1M)
list_devices	割り当て可能なデバイスの状態を表示します。 device_maps ファイルにリストされたデバイスに関連付けられている、デバイス特殊ファイルを列挙します。	list_devices(1)
list_devices -U	割り当て可能なデバイスを一覧表示するか、あるいは特定のユーザー ID に割り当てられているデバイスを一覧表示します。このオプションを使用すると、別のユーザーに割り当てることができるデバイスまたは割り当て済みのデバイスを確認できます。このコマンドを実行するには、ユーザーまたは役割に solaris.device.revoke 承認がなければなりません。	
allocate	1 人のユーザーだけが使用できるように割り当て可能デバイスを予約します。 デフォルトでは、ユーザーはデバイス割り当てを行うのに solaris.device.allocate 承認が必要です。ユーザー承認を必要としないように、device_allocate ファイルを変更することもできます。そのように変更した場合、システム上のどのユーザーでもデバイスの使用割り当てを要求できます。	allocate(1)
deallocate	デバイスから割り当て予約を削除します。	deallocate(1)

割り当てコマンドの承認

デフォルトでは、ユーザーが割り当て可能デバイスを予約するには solaris.device.allocate 承認を必要とします。solaris.device.allocate 承認を含める権利プロファイルを作成する方法については、90 ページの「ユーザーによるデバイス割り当てを承認する方法」を参照してください。

管理者がデバイスの割り当て状態を変更するには、どのデバイスの場合でも、solaris.device.revoke 承認が必要です。たとえば、allocate および list_devices コマンドの -U オプションや、deallocate コマンドの -F オプションは、solaris.device.revoke 承認が必要です。

詳細は、258 ページの「承認を必要とするコマンド」を参照してください。

割り当てエラー状態

deallocate コマンドが割り当ての解除に失敗する場合、または allocate コマンドが割り当てに失敗する場合は、デバイスは「割り当てエラー状態」になります。割り当て可能デバイスが割り当てエラー状態となった場合、そのデバイスの割り当てを

強制的に解除する必要があります。割り当てエラー状態を処理できるのは、スーパーユーザーか、あるいは Device Management 権利プロファイルまたは Device Security 権利プロファイルを持った役割だけです。

F オプションを指定した -deallocate コマンドは、割り当て解除を強制します。あるいは、allocate -U を実行してデバイスを特定のユーザーに割り当てることもできます。いったんデバイスが割り当てられると、発生したエラーメッセージを調査できます。デバイスに関する問題が解決されたあとで、そのデバイスの割り当てを強制的に解除できます。

device_maps ファイル

デバイス割り当てを設定すると、デバイスマップが作成されます。監査サービスが有効になる際には、bsmconv コマンドによってデフォルトの /etc/security/device_maps ファイルが作成されます。この当初の device_maps ファイルは、サイトに合わせてカスタマイズできます。device_maps ファイルには、デバイス名、デバイスの種類のほか、各割り当て可能デバイスに関連付けられたデバイス特殊ファイルが記録されます。

直観的にはわかりにくい各デバイスのために、device_maps ファイルはデバイス特殊ファイルのマッピングを定義します。このファイルによって、プログラムはどのデバイス特殊ファイルがどのデバイスに割り当てられているかを検出できます。たとえば、dminfo コマンドを使用すると、デバイス名、デバイスの種類およびデバイス特殊ファイルを取得して、割り当て可能なデバイスを設定するときに指定できます。dminfo コマンドは、device_maps ファイルを使用してデバイス割り当て情報を報告します。

各デバイスは、次の形式の 1 行のエントリで表されます。

device-name:*device-type*:*device-list*

例 4–14　device_maps エントリの例

次に、フロッピーディスクドライブ fd0 の device_maps ファイル内にあるエントリの例を示します。

```
fd0:\
        fd:\
        /dev/diskette /dev/rdiskette /dev/fd0a /dev/rfd0a \
/dev/fd0b /dev/rfd0b /dev/fd0c /dev/fd0 /dev/rfd0c /dev/rfd0:\
```

device_maps ファイルの行末にバックスラッシュ (\) を付けて、エントリを次の行に続けることができます。コメントも挿入できます。ポンド記号 (#) を付けると、1 つ前の行末にバックスラッシュのない改行まで、それに続くすべてのテキストはコメントになります。どのフィールドでも先行ブランクと後続ブランクを使用できます。フィールドの定義は次のとおりです。

device-name デバイスの名前を指定します。現在のデバイス名の一覧を表示する方法については、91 ページの「デバイスの割り当て情報を表示する方法」を参照してください。

device-type 汎用デバイスタイプを指定します。汎用名は、`st`、`fd`、`audio` などのデバイスクラス名です。*device-type* では、関連するデバイスが論理的にグループ化されます。

device-list 物理デバイスに関連付けられたデバイス特殊ファイルを一覧表示します。*device-list* には、特定のデバイスにアクセスできるすべての特殊ファイルが含まれている必要があります。リストが不完全な場合は、悪意を持ったユーザーでも個人情報を入手または変更できます。*device-list* フィールドには、`/dev` ディレクトリに入っているデバイスファイルを指定します。

device_allocate ファイル

当初の `/etc/security/device_allocate` ファイルは、監査サービスを有効にする際に `bsmconv` コマンドによって作成されます。この当初の `device_allocate` ファイルは、開始点として使用できます。`/etc/security/device_allocate` ファイルを変更して、デバイスを割り当て可能から割り当て不可に変更したり、新しいデバイスを追加したりします。`device_allocate` ファイルの例を次に示します。

```
st0;st;;;;/etc/security/lib/st_clean
fd0;fd;;;;/etc/security/lib/fd_clean
sr0;sr;;;;/etc/security/lib/sr_clean
audio;audio;;;*;/etc/security/lib/audio_clean
```

`device_allocate` ファイル内のエントリは、デバイスが割り当て可能であると特に記述されていない限り、そのデバイスが割り当て可能であることを意味しません。上述の `device_allocate` ファイルの例では、オーディオデバイスエントリの第 5 フィールドにアスタリスク (*) が指定されています。第 5 フィールド内のアスタリスクは、そのデバイスが割り当て可能でないことをシステムに示します。つまり、このデバイスは使用できません。このフィールドにほかの値が入っているか、あるいは値が入っていない場合は、デバイスが使用可能であることを示します。

`device_allocate` ファイルでは、各デバイスは次の形式の 1 行のエントリで表されます。

device-name;*device-type*;`reserved`;`reserved`;*auths*;*device-exec*

`device_allocate` ファイル内の行は、バックスラッシュ (\) で終了することで次の行のエントリに継続できます。コメントも挿入できます。ポンド記号 (#) を付けると、1 つ前の行末にバックスラッシュのない改行まで、それに続くすべてのテキストはコメントになります。どのフィールドでも先行ブランクと後続ブランクを使用できます。フィールドの定義は次のとおりです。

device-name デバイスの名前を指定します。現在のデバイス名の一覧を表示する方法については、91 ページの「デバイスの割り当て情報を表示する方法」を参照してください。

device-type 汎用デバイスタイプを指定します。汎用名は、`st`、`fd`、`sr` などのデバイスクラス名です。*device-type* では、関連するデバイスが論理的にグループ化されます。デバイスを割り当て可能にするときは、`device_maps` ファイルの *device-type* フィールドからデバイス名を取得します。

`reserved` `reserved` で示される 2 つのフィールドは、将来の使用に予約されています。

auths デバイスが割り当て可能かどうかを示します。このフィールドにアスタリスク (*) が入っている場合は、デバイスが割り当て不可能であることを示します。承認を示す文字列が入っている場合や、空の場合は、デバイスが割り当て可能であることを示します。たとえば、*auths* フィールドの文字列 `solaris.device.allocate` は、そのデバイスを割り当てるには `solaris.device.allocate` 承認が必要であることを示します。このフィールドに単価記号 (@) が入っている場合は、どのユーザーでもそのデバイスを割り当てることができることを示します。

device-exec 割り当てプロセス中にクリーンアップやオブジェクト再使用防止などの特殊処理のために呼び出されるスクリプトのパス名を指定します。*device-exec* スクリプトは、デバイスに対して `deallocate` コマンドを実行するたびに実行されます。

たとえば、`sr0` デバイスについての次のエントリは、CD-ROM ドライブが `solaris.device.allocate` 承認を得たユーザーによって割り当て可能であることを示します。

```
sr0;sr;reserved;reserved;solaris.device.allocate;/etc/security/lib/sr_clean
```

デフォルトのデバイスと、その定義済み特性をそのまま使用することもできます。新しいデバイスをインストールしたあとで、エントリを変更できます。使用前に割り当てが必要なデバイスはすべて、そのデバイスのシステムの `device_allocate` ファイルと `device_maps` ファイルで定義する必要があります。現在、カートリッジテープドライブ、フロッピーディスクドライブ、CD-ROM ドライブ、およびオーディオチップが、割り当て可能とみなされます。これらのデバイスタイプには、デバイスクリーンスクリプトが用意されています。

注 – Xylogics テープドライブやアーカイブテープドライブも、SCSI デバイス用の st_clean スクリプトを使用します。モデム、端末、グラフィックスタブレットなどの割り当て可能デバイスについては、独自のデバイスクリーンスクリプトを作成する必要があります。このスクリプトは、対応するデバイスタイプのオブジェクト再使用の要件を満たしている必要があります。

デバイスクリーンスクリプト

デバイス割り当てによって、いわゆるオブジェクト再使用要件の一部が満たされます。「デバイスクリーン」スクリプトは、使用可能なすべてのデータを再使用する前に物理デバイスからパージするというセキュリティー要件に対応するものです。データのクリアは、そのデバイスが別のユーザーによって割り当て可能になる前に実行されます。デフォルトでは、カートリッジテープドライブ、フロッピーディスクドライブ、CD-ROM ドライブ、オーディオデバイスは、デバイスクリーンスクリプトが必要です。Oracle Solaris ではそのスクリプトが提供されます。この節では、デバイスクリーンスクリプトが実行する処理について説明します。

テープ用のデバイスクリーンスクリプト

st_clean デバイスクリーンスクリプトでは、3 つのテープデバイスがサポートされます。

- SCSI ¼ インチテープ
- アーカイブ ¼ インチテープ
- オープンリール ½ インチテープ

st_clean スクリプトでは、mt コマンドの rewoffl オプションを使用してデバイスのクリーンアップを行います。詳細は、mt(1) のマニュアルページを参照してください。このスクリプトは、システムブート中に実行されると、デバイスを照会し、デバイスがオンライン状態であるかどうかを確認します。デバイスがオンラインになっていた場合、スクリプトはさらに、そのデバイスにメディアが挿入されているかどうかを調べます。¼ インチのテープデバイスにメディアが挿入されていた場合、このデバイスは割り当てエラー状態になります。この場合、管理者はそのデバイスを手動でクリーンアップする必要があります。

通常のシステム操作中に、deallocate コマンドを対話型モードで実行すると、メディアを取り出すように求めるプロンプトが表示されます。割り当て解除は、デバイスからメディアが取り出されるまで見送られます。

フロッピーディスクドライブと CD-ROM ドライブ用のデバイスクリーンスクリプト

フロッピーディスクドライブと CD-ROM ドライブ用として、次のデバイスクリーンスクリプトが提供されています。

- **fd_clean** スクリプト – フロッピーディスクドライブ用デバイスクリーンスクリプト。
- **sr_clean** スクリプト – CD-ROM ドライブ用デバイスクリーンスクリプト。

これらのスクリプトは、eject コマンドを使用してドライブからメディアを取り出します。eject コマンドが失敗すると、デバイスは割り当てエラー状態になります。詳細は、eject(1) のマニュアルページを参照してください。

オーディオ用のデバイスクリーンスクリプト

オーディオデバイスは、audio_clean スクリプトを使用してクリーンアップします。スクリプトは、AUDIO_GETINFO ioctl システムコールを実行してデバイスを読み取ります。AUDIO_SETINFO ioctl システムコールを実行してデバイス構成をデフォルトにリセットします。

新しいデバイスクリーンスクリプトの作成

システムに新しく割り当て可能デバイスを追加する場合は、独自のデバイスクリーンスクリプトを作成する必要があります。deallocate コマンドは、デバイスクリーンスクリプトにパラメータを渡します。次に示すように、パラメータはデバイス名を含む文字列です。詳細は、device_allocate(4) のマニュアルページを参照してください。

clean-script `-[I|i|f|S]` *device-name*

デバイスクリーンスクリプトは、成功時には「0」を、失敗時には「0」より大きな値を、それぞれ返す必要があります。オプション -I、 -f、 および -S は、スクリプトの実行モードを決定します。

`-I` システムをブートするときにだけ指定します。すべての出力は、システムコンソールに送られます。失敗した場合や、メディアを強制的に取り出せない場合は、デバイスを割り当てエラー状態にします。

`-i` 出力が抑止される点を除き、-I オプションと同じです。

`-f` 強制的なクリーンアップを行うときに指定します。このオプションは対話型であり、ユーザーがプロンプトに応答するものとみなします。このオプションが付いたスクリプトは、クリーンアップの一部に失敗した場合に、クリーンアップ全体を完了しようとします。

-S　標準クリーンアップを行うときに指定します。このオプションは対話型であり、ユーザーがプロンプトに応答するものとみなします。

◆ ◆ ◆ 第 5 章

基本監査報告機能の使用方法 (作業)

この章では、システム上のファイルの目録を作成する方法と、それらの目録を使用してシステムの整合性をチェックする方法について説明します。基本監査報告機能 (BART) を使用すると、一定期間にわたってシステムのファイルレベルチェックを行い、システムを包括的に検証できます。

この章の内容は次のとおりです。

基本監査報告機能 (概要)

BART は、完全にファイルシステムレベルで稼働するファイル追跡ツールです。BART を使用すると、配備済みのシステムにインストールされているソフトウェアスタックのコンポーネントについての情報をすばやく簡単に、かつ確実に収集できます。このツールには、時間のかかる管理作業を簡易化し、システムネットワークの管理に伴う負担を大幅に減らす効果があります。

BART を使用することで管理者は、既知のベースラインと比較し、ファイルレベルで見てどのような変化がシステムに起きたかを確認できます。BART はベースラインの作成に使用できるほか、インストールと構成がすべて完了しているシステムの目録を制御するためにも使用できます。作成したこのベースラインをあとでシステムのスナップショットと比較すれば、システムのインストール以後にシステムで発生したファイルレベルの変化を示すレポートが生成されます。

`bart` コマンドは、標準の UNIX コマンドです。`bart` コマンドの出力は、あとで処理できるようにファイルにリダイレクトできます。

BART の機能

BART は、強力で柔軟、かつシンプルな構文に重点を置いて設計されています。このツールは、異なる時点で特定のシステムの目録を生成するのに利用できます。システムファイルの検証が必要になった場合には、新旧の目録を比較してレポートを生成できます。また、複数の類似したシステムの目録を生成し、システム同士の比較も行えます。BART と既存の監査ツールの違いは、追跡の対象となる情報と、レポートされる情報の両方に関して BART は柔軟であることです。

BART には、ほかにも次のようなメリットや利用法があります。

- ファイルレベルで Oracle Solaris ソフトウェアを実行しているシステムを効率良く簡単にカタログ化できます。
- どのファイルを監視するかを決定でき、必要に応じてプロファイルの変更も可能です。この柔軟性のおかげでローカルなカスタマイズを監視でき、ソフトウェアの再構成も簡単に効率良く行えます。
- 信頼できるソフトウェアだけをシステムで稼働させることができます。
- 一定期間におけるシステムのファイルレベルの変化を監視できます (ファイルレベルの監視を行うと、破損ファイルや異常なファイルを見つけやすくなる)。
- システムパフォーマンスの問題の解決に役立ちます。

BART コンポーネント

BART には、主要コンポーネントが 2 つ、オプションコンポーネントが 1 つ存在します。これらを次に示します。

- BART 目録
- BART レポート
- BART 規則ファイル

BART 目録

`bart create` コマンドを使用して、特定の時点におけるシステムのファイルレベルスナップショットを作成できます。このコマンドでは、ファイルのカタログと、「目録」と呼ばれるファイル属性が出力されます。目録には、システム上のすべてのファイルまたは特定のファイルについての情報が示されます。これはファイルの属性についての情報が入ったものであり、MD5 チェックサムなどの一意の識別情報も含めることができます。MD5 チェックサムについての詳細は、`md5(3EXT)` のマニュアルページを参照してください。目録は、保存して、クライアントシステムとサーバーシステムの間で転送できます。

注 – ファイルシステムのタイプが同じである場合を除き、BART がファイルシステムの境界を越えることはありません。この制約があるため、bart create コマンドの出力を予測しやすくなります。たとえば、引数を指定せずに bart create コマンドを実行すると、ルート (/) ディレクトリの下のすべてのファイルシステムがカタログ化されます。しかし、NFS ファイルシステム、TMPFS ファイルシステム、マウントされた CD-ROM はカタログ化されません。目録を作成するときは、ネットワーク上のファイルシステムは監査の対象としないでください。ネットワーク化されたファイルシステムを BART で監視すると、リソースを大量に消費し、ほとんど価値のない目録が生成されます。

BART 目録の詳細は、126 ページの「BART 目録のファイル形式」を参照してください。

BART レポート

このレポートツールの入力情報は 3 つあります。 比較される目録 2 つと、ユーザーによって任意に指定される規則ファイル 1 つです。規則ファイルは、どのような相違が監視されるかを指定するものです。

2 つの目録、制御目録とテスト目録の比較には、bart compare コマンドを使用します。これらの目録は、bart create コマンドで使用するものと同じファイルシステム、オプション、および規則ファイルで用意する必要があります。

bart compare コマンドで出力されるのは、ファイルごとに 2 つの目録の相違を示したレポートです。「相違」は、両方の目録のためにカタログ化されている特定のファイルの属性変化です。2 つの目録間でファイルエントリの追加または削除があった場合も相違とみなされます。

相違を報告するときの制御レベルは 2 つあります。

- 目録を生成する時点
- レポートを生成する時点

目録の生成は 2 つの目録間の相違を報告するよりも負担が大きいため、これらのレベルの制御は計画的に行われます。目録の作成が終わると、さまざまな規則ファイルを使用して bart compare コマンドを実行し、異なる視点から目録を比較できます。

BART レポートの詳細は、129 ページの「BART レポート」を参照してください。

BART 規則ファイル

規則ファイルは、bart コマンドへの入力情報として任意に指定できるテキストファイルです。このファイルは、取り込みについての規則と除外についての規則を使用します。規則ファイルは、カスタム目録とカスタムレポートの作成に使用され

ます。このファイルには、どのファイル群をカタログ化するか、特定のファイル群のどの属性を監視するかを指定できます。目録を比較する場合、目録間の相違を報告するには規則ファイルが役立ちます。規則ファイルを使用することで、システム上のファイルについての特定の情報を効率良く収集できます。

規則ファイルは、テキストエディタで作成できます。次に、規則ファイルを使用して行える作業を示します。

- `bart create` コマンドを使用し、システム上の全ファイルまたは特定のファイルについての情報を列挙する目録を作成します。
- `bart compare` コマンドを使用し、ファイルシステムの特定の属性を監視するレポートを生成します。

注 – 目的に合わせて、複数の複数の規則ファイルを作成できます。しかし、規則ファイルを使用して目録を作成する場合は、それらの目録を比較する際に同じ規則ファイルを使用する必要があります。規則ファイルを使用して作成された目録を比較する時に同じ規則ファイルを使用しないと、`bart compare` コマンドは不正な相違を多数表示します。

規則ファイルには、ユーザーエラーの結果として構文エラーなどのあいまいな情報も含めることができます。規則ファイルに誤った情報が含まれている場合は、それらのユーザーエラーも報告されます。

規則ファイルを使用してシステム上の特定のファイルやファイル属性を監視する場合は、計画が必要です。規則ファイルを作成する前に、システム上のどのファイルとファイル属性を監視するかを決定してください。何を達成するかに基づき、目録の作成、目録の比較、あるいはその他の目的に規則ファイルを利用できます。

BART 規則ファイルの詳細は、128 ページの「BART 規則ファイルの書式」と、`bart_rules(4)` のマニュアルページを参照してください。

BARTの使用方法(作業マップ)

作業	説明	参照先
BART 目録を作成します。	システムにインストールされているファイルごとに情報の一覧表示を生成します。	114 ページの「目録を作成する方法」

作業	説明	参照先
カスタム BART 目録を作成します。	システムにインストールされている特定のファイルについての情報の一覧を、次に示す方法のいずれかで生成します。 ■ サブツリーを指定する ■ ファイル名を指定する ■ 規則ファイルを使用する	116 ページの「目録をカスタマイズする方法」 例 5–2 例 5–3 例 5–4
BART 目録を比較します。	一定期間における同一システムの変化を比較するレポートを生成します。 あるいは、1 つまたは複数のシステムを制御システムと比較するレポートを生成します。	119 ページの「一定期間内で同一システムの目録を比較する方法」 121 ページの「異なるシステムからの目録の比較方法」
(省略可能) BART レポートをカスタマイズします。	次に示す方法のいずれかでカスタム BART レポートを生成します。 ■ 属性を指定する ■ 規則ファイルを使用する	124 ページの「ファイル属性を指定して BART レポートをカスタマイズする方法」 125 ページの「規則ファイルを使用して BART レポートをカスタマイズする方法」

BART の使用方法 (作業)

bart コマンドは、通常のユーザーとしても、スーパーユーザーとしても、あるいは Primary Administrator 役割を引き受けたユーザーとしても実行できます。ただし、bart コマンドを通常のユーザーとして実行する場合は、アクセス権のあるファイルのカタログ化と監視しか行えません (たとえば、自分のホーム内のファイルのみ)。スーパーユーザーとして bart コマンドを実行する利点は、作成する目録に隠しファイルやプライベートファイルについての情報が含まれることです。これらの情報を監視することもあるでしょう。アクセス権を制限したファイル (/etc/passwd ファイルや /etc/shadow ファイル) に関する情報のカタログ化と監視が必要な場合は、bart コマンドをスーパーユーザーとして実行してください。RBAC の使用についての詳細は、188 ページの「役割によるアクセス制御 (概要)」を参照してください。

BART におけるセキュリティー上の考慮事項

bart コマンドをスーパーユーザーとして実行した場合、その出力は誰にでも読み取れます。この出力には、プライベートであることを意図するファイル名も含まれる可能性があります。bart コマンドの実行時にスーパーユーザーになる場合には、出力を適切に保護してください。たとえば、アクセス権を制限した状態で出力ファイルを生成するオプションを使用します。

注 – この章に示されている作業と例は、スーパーユーザーによって実行された bart コマンドを示しています。特に明記しない限り、bart コマンドをスーパーユーザーとして実行するかどうかは任意に選択できます。

▼ 目録を作成する方法

システムの目録は、Oracle Solaris ソフトウェアの初期インストールが終わった直後に作成できます。このタイプの目録は、一定期間における同一システムの変化を比較するためのベースラインとなります。あるいは、この目録を使用し、異なるシステムの目録と比較することもできます。たとえば、ネットワーク上の各システムのスナップショットを作成し、各テスト目録を制御目録と比較する場合、テストシステムをベースライン構成と一致させるために必要な作業をすばやく判断できます。

1 **Primary Administrator 役割を引き受けるか、スーパーユーザーになります。**

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 **Oracle Solaris ソフトウェアのインストール後に制御目録を作成し、その出力をファイルにリダイレクトします。**

```
# bart create options > control-manifest
```

`-R` 目録の検査対象としてルートディレクトリを指定します。規則で指定されるパスはすべて、このディレクトリからの相対パスとなるように変換されます。目録で報告されるパスはすべて、このディレクトリからの相対パスとなります。

`-I` カタログ化される個々のファイルの一覧 (コマンド行上で指定されるか、あるいは標準入力から読み取られる) を受け入れます。

`-r` この目録の規則ファイルの名前です。- を付けて -r オプションを使用すると、規則ファイルが標準入力から読み取られます。

`-n` ファイルリストに挙げられたすべての通常ファイルの署名を無効にします。このオプションは、パフォーマンスを上げる目的で使用できます。また、(システムログファイルの場合のように) ファイルリストの内容が変わる予定がある場合にも使用できます。

3 目録の内容を確認します。

4 あとで利用できるように目録を保存します。

目録にわかりやすい名前をつけてください。たとえば、システム名と目録が作成された日付を組み合わせます。

例 5–1 システム上のファイルごとに情報を一覧表示する目録を作成する

オプションをまったく指定せずに bart create コマンドを実行すると、システムにインストールされているファイルごとに情報がカタログ化されます。このタイプの目録は、元になるイメージから多数のシステムをインストールする場合のベースラインとして使用してください。また、同一のインストールが行われたかを確認する場合に、このタイプの目録を使用して比較を行うこともできます。

次に例を示します。

```
# bart create
! Version 1.0
! Thursday, December 04, 2003 (16:17:39)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/ D 1024 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3fd9ea47 0 0
/.java D 512 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3f8dc04d 0 10
/.java/.userPrefs D 512 40700 user::rwx,group::---,mask:---
other:--- 3f8dc06b 010
/.java/.userPrefs/.user.lock.root F 0 100600 user::rw-
group::---,mask:---,other:--- 3f8dc06b 0 10 -
/.java/.userPrefs/.userRootModFile.root F 0 100600 user::rw-,
group::---,mask:---,other:--- 3f8dc0a1 0 10 -
.
.
.
/var/sadm/pkg/SUNWdtmad/install/depend F 932 100644 user::rw-,
group::r--,mask:r--,other:r-- 3c23a19e 0 0 -
/var/sadm/pkg/SUNWdtmad/pkginfo F 594 100644 user::rw-
group::r--,mask:r--,other:r-- 3f81e416 0 0 -
/var/sadm/pkg/SUNWdtmad/save D 512 40755 user::rwx,group::r-x
mask:r-x,other:r-x 3f81e416 0 0
/var/sadm/pkg/SUNWdtmaz D 512 40755 user::rwx,group::r-x
mask:r-x,other:r-x 3f81e41b 0 0
/var/sadm/pkg/TSIpgxw/save D 512 40755 user::rwx
group::r-x,mask:r-x,other:r-x 3f81e892 0 0
.
.
.
```

各目録は、ヘッダーとエントリから構成されます。目録ファイルの各エントリは、ファイルタイプに応じて単一の行に示されます。たとえば、上記の出力における各目録エントリでは、タイプ F はファイルを示し、タイプ D はディレクトリを示します。また、サイズ、内容、ユーザー ID、 グループ ID、 およびアクセス権も表示されます。特殊文字を正しく処理するため、出力内のファイルエントリはコード化されたバージョンのファイル名でソートされます。この結果、すべてのエントリが

ファイル名をキーにして昇順に並べられます。標準でないファイル名(改行文字やタブ文字が埋め込まれたものなど)については、ソート前にそれらの非標準文字が引用符で囲まれます。

! で始まる行には、目録についてのメタデータが示されてます。目録バージョン行には、目録の仕様バージョンが示されます。日付行には、目録が作成された日付が日付形式で示されます。date(1) のマニュアルページを参照してください。一部の行は、目録比較ツールによって無視されます。無視される行には、空の行や、空白しか入っていない行、# から始まるコメントなどがあります。

▼ 目録をカスタマイズする方法

目録は、次に示す方法のいずれかでカスタマイズできます。

- サブツリーを指定する

  システム上の個々のサブツリーに目録を 1 つ作成するという方法を採ると、大きなディレクトリのすべての内容にそれぞれ目録を作成するよりも、特定のファイルの変化を効率良く監視できます。この方法では、システム上の特定のサブツリーのベースライン目録を作成し、その後そのサブツリーのテスト目録を定期的に作成できます。制御目録とテスト目録の比較には、`bart compare` コマンドを使用します。このオプションを使用すると、重要なファイルシステムを効率良く監視し、侵入者によって攻撃されたファイルがないかを確認できます。

- ファイル名を指定する

  全システムをカタログ化する目録の作成は時間がかかり、ディスクスペースや負担も大きくなるため、システム上の特定のファイルまたは特定のファイル群についての情報だけを表示する場合は、この `bart` コマンドオプションを使用することを推奨します。

- 規則ファイルを使用する

  規則ファイルは、単一のシステム上の特定のファイルと特定のサブツリーに関する情報を表示するカスタム目録を作成する場合に利用できます。規則ファイルは、特定のファイル属性の監視にも使用できます。規則ファイルを使用して目録の作成と比較を行うと、複数のファイルまたはサブツリーの複数の属性を指定するという柔軟さが得られます。しかし、コマンド行では、作成する各目録または生成するレポートの全ファイルに適用される汎用的な属性定義しか指定できません。

1 どのファイルのカタログ化および監視を行うのかを決定します。

2 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

3 **Oracle Solaris** ソフトウェアのインストール後、次に示すオプションのいずれかを使用してカスタム目録を作成します。

- サブツリーを指定する。

  ```
  # bart create -R root-directory
  ```

- ファイル名 (1 つまたは複数) を指定する。

  ```
  # bart create -I filename...
  ```

  次に例を示します。

  ```
  # bart create -I /etc/system /etc/passwd /etc/shadow
  ```

- 規則ファイルを使用する。

  ```
  # bart create -r rules-file
  ```

4 目録の内容を確認します。

5 あとで利用できるように目録を保存します。

例 5–2 サブツリーを指定して目録を作成する

この例は、`/etc/ssh` サブツリーだけに含まれるファイルの情報が入った目録を作成する方法を示しています。

```
# bart create -R /etc/ssh
! Version 1.0
! Saturday, November 29, 2003 (14:05:36)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/ D 512 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3f81eab9 0 3
/ssh_config F 861 100644 user::rw-,group::r--,mask:r--,
other:r-- 3f81e504 0 3 422453ca0e2348cd9981820935600395
/ssh_host_dsa_key F 668 100600 user::rw-,group::---,mask:---,
other:--- 3f81eab9 0 0 5cc28cdc97e833069fd41ef89e4d9834
/ssh_host_dsa_key.pub F 602 100644 user::rw-,group::r--,mask:r--,
other:r-- 3f81eab9 0 0 16118c736995a4e4754f5ab4f28cf917
/ssh_host_rsa_key F 883 100600 user::rw-,group::---,mask:---,
other:--- 3f81eaa2 0 0 6ff17aa968ecb20321c448c89a8840a9
```

```
/ssh_host_rsa_key.pub F 222 100644 user::rw-,group::r--,mask:r--,
other:r-- 3f81eaa2 0 0 9ea27617efc76058cb97aa2caa6dd65a
.
.
.
```

例 5–3 ファイル名を指定することによって目録をカスタマイズする

この例は、システム上の /etc/passwd ファイルと /etc/shadow ファイルについての情報だけを表示する目録を作成する方法を示しています。

```
# bart create -I /etc/passwd /etc/shadow
! Version 1.0
! Monday, December 15, 2003 (16:28:55)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/etc/passwd F 542 100444 user::r--,group::r--,mask:r--,
other:r-- 3fcfd45b 0 3 d6
84554f85d1de06219d80543174ad1a
/etc/shadow F 294 100400 user::r--,group::---,mask:---,
other:--- 3f8dc5a0 0 3 fd
c3931c1ae5ee40341f3567b7cf15e2
```

比較として、同じシステム上の /etc/passwd ファイルと /etc/shadow ファイルに対して ls -al コマンドを実行した場合の標準出力を次に示します。

```
# ls -al /etc/passwd
-r--r--r--   1 root     sys          542 Dec  4 17:42 /etc/passwd

# ls -al /etc/shadow
-r--------   1 root     sys          294 Oct 15 16:09 /etc/shadow
```

例 5–4 規則ファイルを使用して目録をカスタマイズする

この例は、/etc ディレクトリ内のファイルだけをカタログ化するために規則ファイルを使用して目録を作成する方法を示しています。この規則ファイルには、/etc/system ファイルの acl 属性の変化を監視するために bart compare コマンドによって使用される指示語も含まれます。

- テキストエディタを使用し、/etc ディレクトリ内のファイルだけをカタログ化する規則ファイルを作成します。

  ```
  # List information about all the files in the /etc directory.

  CHECK all
  /etc
  ```

```
# Check only acl changes in the /etc/system file

IGNORE all
CHECK acl
/etc/system
```

規則ファイルの作成についての詳細は、111 ページの「BART 規則ファイル」を参照してください。

- 作成した規則ファイルを使用して制御目録を作成します。

```
# bart create -r etc.rules-file > etc.system.control-manifest
! Version 1.0
! Thursday, December 11, 2003 (21:51:32)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/etc/system F 1883 100644 user::rw-,group::r--,mask:r--,
other:r-- 3f81db61 0 3
```

- システムに発生した変化を監視する時点でテスト目録を作成します。このテスト目録は、同じ bart オプションと同じ規則ファイルを使って制御目録と同じように用意してください。
- 同じ規則ファイルを使用し、目録を比較します。

▼ 一定期間内で同一システムの目録を比較する方法

この作業は、一定期間内で同一のシステムに起きるファイルレベルの変化を監視する場合に行なってください。このタイプの目録は、セキュリティー侵害を検出して破損したファイルや異常な状態のファイルを見つけたり、システムのパフォーマンスの問題を解決したりするのに便利です。

1 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 **Oracle Solaris** ソフトウェアのインストール後、システム上で監視するファイルの制御目録を作成します。

```
# bart create -R /etc > control-manifest
```

3 システムの変化を監視する任意の時点で、制御目録と同様の指定でテスト目録を作成します。

```
# bart create -R /etc > test-manifest
```

4 制御目録をテスト目録と比較します。

```
# bart compare options control-manifest test-manifest > bart-report
```

`-r`	この比較の規則ファイルの名前です。`--` を付けて `r` オプションを使用するのは、指示語が標準入力から読み取られることを意味します。
`-i`	ユーザーは、コマンド行から汎用指示語 `IGNORE` を設定できます。
`-p`	プログラムによる解析に対して地域対応化されていない標準的な出力を生成するプログラムモードです。
control-manifest	制御システムの `bart create` コマンドからの出力です。
test-manifest	テストシステムの `bart create` コマンドからの出力です。

5 **BART** レポートを調べ、異常がないかを確認します。

例 5–5 同一システムの目録を一定期間で比較する

この例では、一定期間に `/etc` ディレクトリに発生した変化を監視します。このタイプの比較を行うと、システム上の重要ファイルが攻撃されていないかをすばやく確認できます。

- 制御目録を作成します。

```
# bart create -R /etc > system1.control.121203
! Version 1.0
! Friday, December 12, 2003 (08:34:51)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/ D 4096 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3fd9dfb4 0 3
/.cpr_config F 2236 100644 user::rw-,group::r--,mask:r--,other:r--
3fd9991f 0 0
67cfa2c830b4ce3e112f38c5e33c56a2
/.group.lock F 0 100600 user::rw-,group::---,mask:---,other:--- 3f81f14d
0 1 d41
d8cd98f00b204e9800998ecf8427e
/.java D 512 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3f81dcb5 0 2
/.java/.systemPrefs D 512 40755 user::rwx,group::r-x,mask:r-x,
other:r-x 3f81dcb7
.
```

```
.
.
```

- /etc ディレクトリの変化を監視する時点で、テスト目録を作成します。

```
# bart create -R /etc > system1.test.121503
Version 1.0
! Monday, December 15, 2003 (08:35:28)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/ D 4096 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3fd9dfb4 0 3
/.cpr_config F 2236 100644 user::rw-,group::r--,mask:r--,other:r--
3fd9991f 0 0
67cfa2c830b4ce3e112f38c5e33c56a2
/.group.lock F 0 100600 user::rw-,group::---,mask:---,other:---
3f81f14d 0 1 d41d8cd98f00b204e9800998ecf8427e
/.java D 512 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3f81dcb5 0 2
/.java/.systemPrefs D 512 40755 user::rwx,group::r-x,mask:r-x,
other:r-x 3f81dcb70 2
/.java/.systemPrefs/.system.lock F 0 100644 user::rw-,group::r--
,mask:r--,other:
r-- 3f81dcb5 0 2 d41d8cd98f00b204e9800998ecf8427e
/.java/.systemPrefs/.systemRootModFile F 0 100644 user::rw-,
group::r--,mask:r--,
other:r-- 3f81dd0b 0 2 d41d8cd98f00b204e9800998ecf8427e
.
.
.
```

- 制御目録をテスト目録と比較します。

```
# bart compare system1.control.121203 system1.test.121503
/vfstab:
  mode  control:100644  test:100777
  acl  control:user::rw-,group::r--,mask:r--,other:r--  test:user::rwx,
group::rwx,mask:rwx,other:rwx
```

上記の出力は、制御目録が作成されたあとに vfstab ファイルのアクセス権が変化したことを示しています。このレポートは、所有権、日付、内容などのファイル属性が変化していないかを確認するために使用できます。このタイプの情報をすぐに利用できるようにしておくと、ファイルを改ざんした可能性がある人物や、変化が起きた時点などを追跡しやすくなります。

▼ 異なるシステムからの目録の比較方法

システム同士の比較を行い、ベースラインシステムとほかのシステムの間にファイルレベルの相違がないかをすばやく確認できます。たとえば、ベースラインシステムに特定バージョンの Oracle Solaris ソフトウェアをインストールした場合で、ほか

のシステムにも同一のパッケージがインストールされているかどうかを知りたいときには、それらのシステムの目録を作成し、続いてテスト目録を制御目録と比較できます。このタイプの比較では、制御システムと比較される各テストシステムについてファイルの内容の相違がすべて一覧表示されます。

1 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 **Oracle Solaris** ソフトウェアのインストール後、制御目録を作成します。

```
# bart create options > control-manifest
```

3 制御目録を保存します。

4 テストシステムで、同じ **bart** オプションを使用して目録を作成し、出力をファイルにリダイレクトします。

```
# bart create options > test1-manifest
```

テスト目録に識別しやすい意味のある名前を付けます。

5 目録を比較する準備が整うまで、このテスト目録をシステムの中心的な場所に保存しておきます。

6 目録を比較する時点で、制御目録をテスト目録の場所へコピーするか、あるいはテスト目録を制御システムへコピーします。

次に例を示します。

```
# cp control-manifest /net/test-server/bart/manifests
```

テストシステムが NFS マウントされた状態でない場合は、FTP などの信頼性のある方法によって制御目録をテストシステムにコピーしてください。

7 制御目録をテスト目録と比較し、出力をファイルにリダイレクトします。

```
# bart compare control-manifest test1-manifest > test1.report
```

8 **BART** レポートを調べ、異常がないかを確認します。

9 制御目録と比較を行いたいテスト目録ごとに手順 **4** から **9** を繰り返します。

テストシステムごとに、同じ `bart` オプションを使用してください。

例 5–6 異なるシステムの目録を制御システムの目録と比較する

この例では、制御目録を異なるシステムのテスト目録と比較することによって /usr/bin ディレクトリの内容に起きた変化を監視する方法について説明します。

- 制御目録を作成します。

```
# bart create -R /usr/bin > control-manifest.121203
!Version 1.0
! Friday, December 12, 2003 (09:19:00)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/ D 13312 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3fd9e925 0 2
/.s F 14200 104711 user::rwx,group::--x,mask:--x,other:--x
3f8dbfd6 0 1 8ec7e52d8a35ba3b054a6394cbf71cf6
/ControlPanel L 28 120777 - 3f81dc71 0 1 jre/bin/ControlPanel
/HtmlConverter L 25 120777 - 3f81dcdc 0 1 bin/HtmlConverter
/acctcom F 28300 100555 user::r-x,group::r-x,mask:r-x,other:r-x
3f6b5750 0 2 d6e99b19c847ab4ec084d9088c7c7608
/activation-client F 9172 100755 user::rwx,group::r-x,mask:r-x,
other:r-x 3f5cb907 0 1 b3836ad1a656324a6e1bd01edcba28f0
/adb F 9712 100555 user::r-x,group::r-x,mask:r-x,other:r-x
3f6b5736 0 2 5e026413175f65fb239ee628a8870eda
/addbib F 11080 100555 user::r-x,group::r-x,mask:r-x,other:r-x
3f6b5803 0 2 a350836c36049febf185f78350f27510
.
.
.
```

- 制御システムと比較するシステムごとにテスト目録を作成します。

```
# bart create -R /usr/bin > system2-manifest.121503
! Version 1.0
! Friday, December 15, 2003 (13:30:58)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/ D 13312 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3fd9ea9c 0 2
/.s F 14200 104711 user::rwx,group::--x,mask:--x,other:--x
3f8dbfd6 0 1 8ec7e52d8a35ba3b054a6394cbf71cf6
/ControlPanel L 28 120777 - 3f81dc71 0 1 jre/bin/ControlPanel
/HtmlConverter L 25 120777 - 3f81dcdc 0 1 bin/HtmlConverter
/acctcom F 28300 100555 user::r-x,group::r-x,mask:r-x,other:
r-x 3f6b5750 0 2 d6e99b19c847ab4ec084d9088c7c7608
.
.
.
```

- 目録を比較する時点で、その目録を同じ場所にコピーします。

```
# cp control-manifest /net/system2.central/bart/manifests
```

- 制御目録をテスト目録と比較します。

```
# bart compare control-manifest  system2.test > system2.report
/su:
  gid  control:3  test:1
/ypcat:
  mtime  control:3fd72511  test:3fd9eb23
```

上記の出力は、/usr/binディレクトリのsuファイルのグループIDが制御システムのグループIDと同じでないことを示しています。この情報は、異なるバージョンのソフトウェアがテストシステムにインストールされていないかということや、誰かがファイルを改ざんしていないかということなどの確認に便利です。

▼ ファイル属性を指定してBARTレポートをカスタマイズする方法

ここでは、コマンド行でファイル属性を指定することによってBARTレポートをカスタマイズする方法を説明します。この作業はオプション(省略可能)です。システム上の全ファイルまたは特定のファイルについての情報を一覧表示するベースライン目録を作成すると、特定のディレクトリ、サブディレクトリ、またはファイル(1つまたは複数)に起きた変化を監視する任意の時点で各種の属性を指定してbart compareコマンドを実行できます。コマンド行で各種のファイル属性を指定し、同一目録についてさまざまな比較を行うことができます。

1. 監視するファイル属性を決定します。
2. **Primary Administrator**役割を引き受けるか、スーパーユーザーになります。

   Primary Administrator役割には、Primary Administratorプロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solarisのシステム管理(基本編)』の第2章「Solaris管理コンソールの操作(手順)」を参照してください。
3. **Oracle Solaris**ソフトウェアのインストール後、制御目録を作成します。
4. 変化を監視する時点で、テスト目録を作成します。

   このテスト目録は制御目録と同様に作成してください。
5. 目録を比較します。

   次に例を示します。

```
# bart compare -i dirmtime,lnmtime,mtime control-manifest.121503 \
test-manifest.010504 > bart.report.010504
```

コンマは、コマンド行構文で指定する各属性を区切るために使用します。

6 **BART** レポートを調べ、異常がないかを確認します。

▼ 規則ファイルを使用して **BART** レポートをカスタマイズする方法

ここでは、`bart compare` コマンドの入力情報として規則ファイルを指定することにより BART レポートをカスタマイズする方法を説明します。この作業もオプション (省略可能) です。規則ファイルを使用すると、BART レポートをカスタマイズし、1 つ以上のファイルまたはサブツリーの複数の属性を柔軟に指定できます。また、異なる複数の規則ファイルを使用することで、同じ目録についてさまざまな比較を行えます。

1 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 監視するファイルとファイル属性を決定します。

3 テキストエディタを使用し、適切な指示語を指定して規則ファイルを作成します。

4 **Oracle Solaris** ソフトウェアのインストール後、作成済みの規則ファイルを使用して制御目録を作成します。

```
# bart create -r rules-file > control-manifest
```

5 制御目録と同様に指定してテスト目録を作成します。

```
# bart create -r rules-file > test-manifest
```

6 同じ規則ファイルを使用して、制御目録をテスト目録と比較します。

```
# bart compare -r rules-file control-manifest test-manifest > bart.report
```

7 **BART** レポートを調べ、異常がないかを確認します。

例 5–7 規則ファイルを使用して BART レポートをカスタマイズする

次に示す規則ファイルには、`bart create` コマンドと `bart compare` コマンドの両方に適用される指示語が含まれています。この規則ファイルは、`/usr/bin` ディレクトリ

の内容についての情報を一覧表示するように bart create コマンドに指示します。さらに、このディレクトリのサイズと内容の変化だけを追跡するように bart compare コマンドに指示します。

```
# Check size and content changes in the /usr/bin directory.
# This rules file only checks size and content changes.
# See rules file example.

IGNORE all
CHECK size contents
/usr/bin
```

- 作成した規則ファイルを使用して制御目録を作成します。

  ```
  # bart create -r bartrules.txt > usr_bin.control-manifest.121003
  ```

- /usr/bin ディレクトリの変化を監視する時点で、テスト目録を作成します。

  ```
  # bart create -r bartrules.txt > usr_bin.test-manifest.121103
  ```

- 同じ規則ファイルを使用して目録を比較します。

  ```
  # bart compare -r bartrules.txt usr_bin.control-manifest \
  usr_bin.test-manifest
  ```

- bart compare コマンドの出力を調べます。

  ```
   /usr/bin/gunzip:  add
  /usr/bin/ypcat:
    delete
  ```

上記の出力では、bart compare コマンドによって /usr/bin ディレクトリにおける相違が報告されます。この出力は、/usr/bin/ypcat ファイルが削除されたことと、/usr/bin/gunzip ファイルが追加されたことを示しています。

BART 目録、規則ファイル、およびレポート (参照)

この節では次の参考情報を示します。

- 126 ページの「BART 目録のファイル形式」
- 128 ページの「BART 規則ファイルの書式」
- 129 ページの「BART レポート」

この節では、BART が使用して作成するファイルの形式について説明します。

BART 目録のファイル形式

目録ファイルの各エントリは、ファイルタイプに応じて単一の行に示されます。各エントリは、ファイルの名前である *fname* で始まります。ファイル名に使用された

特殊文字が引き起こす解析上の問題を防ぐため、ファイル名はコード化されます。詳細は、128 ページの「BART 規則ファイルの書式」を参照してください。

後続のフィールドは、次のファイル属性を表します。

type ファイルの種類であり、次のような値となります。

- B: ブロックデバイスノード
- C: キャラクタデバイスノード
- D: ディレクトリ
- F: ファイル
- L: シンボリックリンク
- P: パイプ
- S: ソケット

size ファイルサイズ (バイト)。

mode ファイルのアクセス権を示す 8 進の数値。

acl ファイルの ACL 属性。ACL 属性を持つファイルの場合、`acltotext()` の出力が入ります。

uid このエントリの所有者の、数値で示したユーザー ID。

gid このエントリの所有者の、数値で示したグループ ID。

dirmtime ディレクトリに最後に変化が起きた時点。1970 年 1 月 1 日の 00:00:00 UTC から数えた秒数で示されます。

lnmtime リンクに最後に変化が起きた時点。1970 年 1 月 1 日の 00:00:00 UTC から数えた秒数で示されます。

mtime ファイルに最後に変化が起きた時点。1970 年 1 月 1 日の 00:00:00 UTC から数えた秒数で示されます。

contents ファイルのチェックサム値。この属性が指定されるのは通常ファイルのみです。コンテキストのチェックを無効にした場合と、チェックサムが計算できない場合は、このフィールドの値は – になります。

dest シンボリックリンクのリンク先。

devnode デバイスノードの値。この属性が使用されるのは、キャラクタデバイスファイルとブロックデバイスファイルのみです。

BART 目録についての詳細は、`bart_manifest(4)` のマニュアルページを参照してください。

BART 規則ファイルの書式

bart コマンドの入力ファイルはテキストファイルです。これらのファイルは、目録に含められるファイルと、レポートに含められるファイル属性を指定する行から構成されます。この同じ入力ファイルは、両方の BART 機能で使用できます。#、 空の行、空白を含む行は、ツールが無視します。

入力ファイルには、次に示す 3 種類の指示語が指定されます。

- オプションのパターンマッチング修飾子を持つ subtree 指示語
- CHECK 指示語
- IGNORE 指示語

例 5–8 規則ファイルの書式

```
<Global CHECK/IGNORE Directives>
<subtree1> [pattern1..]
<IGNORE/CHECK Directives for subtree1>

<subtree2> [pattern2..]
<subtree3> [pattern3..]
<subtree4> [pattern4..]
<IGNORE/CHECK Directives for subtree2, subtree3, subtree4>
```

注 – すべての指示語は指定された順に読み取られますが、あとから指定された指示語が先に指定された指示語に優先して読み取られる可能性があります。

行ごとに subtree 指示語が 1 つ存在します。この指示語は、絶対パス名で始まり、そのあとに 0 個以上のパターンマッチング文が続く必要があります。

規則ファイルの属性

bart コマンドは、CHECK 文と IGNORE 文を使用して追跡または無視の対象となる属性を定義します。各属性にはキーワードが関連付けられます。

属性「キーワード」を次に示します。

- acl
- all
- contents
- dest
- devnode
- dirmtime
- gid
- lnmtime

- モード
- mtime
- size
- type
- uid

キーワード all は、すべてのファイル属性を意味します。

引用構文

BART が規則ファイルに使用する記述言語は、標準に準拠していないファイル名を表現する標準の UNIX 引用構文です。埋め込まれたタブ、スペース、改行、特殊文字は、ツールがファイル名を読み取ることができるようにそれらの 8 進形式にコード化されます。この変動的な引用構文では、埋め込みのキャリッジリターンを含むファイル名などがコマンドパイプラインで正しく処理されません。規則記述言語を使用することで、シェル構文だけでは表現が難しい効率の悪い複雑なファイル名フィルタリング基準を表現できます。

BART 規則ファイルや、BART で使用される引用構文についての詳細は、bart_rules(4) のマニュアルページを参照してください。

BART レポート

デフォルトモードでは、次の例に示すように bart compare コマンドは、ディレクトリ変更のタイムスタンプ (dirmtime) を除きシステムにインストールされているすべてのファイルをチェックします。

```
CHECK all
IGNORE    dirmtime
```

規則ファイルを指定すると、汎用指示語である CHECK all と IGNORE dirmtime がこの順で規則ファイルの先頭に自動的に付けられます。

BART の出力

次の終了値が返されます。

0　成功

1　ファイル処理時の致命的でないエラー (アクセス権問題など)

>1　致命的なエラー (無効なコマンド行オプションなど)

レポーティングメカニズムとして、 詳細出力と、プログラムを考慮した出力の 2 種類を利用できます。

- 詳細出力はデフォルトの出力であり、地域対応化され、複数の行にわたって表示されます。詳細出力は国際化されたもので、ユーザーが内容を読み取ることができます。`bart compare` コマンドは、2 つのシステム目録を比較する時に、ファイル間の相違を示すリストを生成します。

  次に例を示します。

  filename attribute control:xxxx test:yyyy

  filename 制御目録とテスト目録で相違があるファイルの名前。

  attribute 比較された目録間で相違があるファイル属性の名前。*xxxx* は制御目録の属性値で、*yyyy* はテスト目録の属性値です。同じファイルの複数の属性に相違が見つかった場合、別々の行にそれぞれの相違が示されます。

  次に、`bart compare` コマンドのデフォルト出力の例を示します。この例では、`/etc/passwd` ファイルでの属性の相違がチェックされています。この出力から、`size`、 `mtime`、 および `contents` 属性に変化があったことがわかります。

```
/etc/passwd:
size     control:74     test:81
mtime control:3c165879     test:3c165979
contents     control:daca28ae0de97afd7a6b91fde8d57afa
test:84b2b32c4165887355317207b48a6ec7
```

- `-bart compare` コマンドの実行時に `p` オプションを指定すると、プログラムを考慮した出力が生成されます。この出力は、プログラム化された操作に適したフォームで生成されます。「プログラムを考慮した出力」はほかのプログラムで簡単に解析が可能であり、ほかのツールに対する入力情報として使用されるように設計されています。

  次に例を示します。

  filename attribute control-val test-val [*attribute control-val test-val*]*

  filename デフォルト形式における *filename* 属性に同じ

  attribute control-val test-val 各ファイルの制御目録とテスト目録間で異なるファイル属性の説明

`bart` コマンドでサポートされる属性の一覧は、128 ページの「規則ファイルの属性」を参照してください。

BART の詳細は、`bart(1M)` のマニュアルページを参照してください。

◆ ◆ ◆ 第 6 章

ファイルアクセスの制御 (作業)

この章では、Oracle Solaris でファイルを保護する方法について説明します。また、システムに悪影響を与える可能性のあるアクセス権が設定されたファイルからシステムを保護する方法についても説明します。

注 – アクセス制御リスト (ACL) を使って ZFS ファイルを保護する方法については、『Oracle Solaris ZFS 管理ガイド』の第 8 章「ACL および属性を使用した Oracle Solaris ZFS ファイルの保護」を参照してください。

この章の内容は次のとおりです。

UNIX アクセス権によるファイル保護

ファイルは、UNIX ファイルアクセス権と ACL を通して保護することができます。スティッキービットが設定されたファイルと、実行可能なファイルは、特殊なセキュリティー対策が必要です。

ファイルの監視と保護を行うコマンド

この表は、ファイルとディレクトリの監視と保護を行うコマンドについて説明したものです。

表 6–1 ファイルとディレクトリの保護を行うコマンド

コマンド	説明	マニュアルページ
`ls`	ディレクトリ内のファイルとファイル情報を表示します。	`ls(1)`
`chown`	ファイルの所有権を変更します。	`chown(1)`
`chgrp`	ファイルのグループ所有権を変更します。	`chgrp(1)`
`chmod`	ファイルのアクセス権を変更します。記号モード (文字と記号) または絶対モード (8 進数) を使用して、ファイルのアクセス権を変更できます。	`chmod(1)`

ファイルとディレクトリの所有権

従来の UNIX ファイルアクセス権では、次に示す 3 つのユーザークラスに所有権を割り当てることができます。

- ユーザー – ファイルまたはディレクトリの所有者。通常はファイルを作成したユーザーです。ファイルの所有者は、そのファイルの読み取り、書き込み (ファイルを変更する)、 または実行 (コマンドの場合) が行えるユーザーを決定できます。
- グループ – ユーザーグループのメンバー。
- その他 – ファイル所有者ではなくグループメンバーでもないその他の全ユーザー。

ファイルアクセス権の割り当てや変更が行えるのは、通常、そのファイルの所有者だけです。しかし、ファイルの所有権の変更は、管理権限のあるユーザー (スーパーユーザーなど) または役割 (Primary Administrator 役割など) によっても行えます。システムポリシーを上書きする方法については、例 6–2 を参照してください。

ファイルは、7 つの形式のいずれかになります。各形式は、次のように記号で示されます。

- (マイナス記号)	テキストまたはプログラム
b	ブロック型特殊ファイル
c	文字型特殊ファイル
d	ディレクトリ
l	シンボリックリンク
s	ソケット
D	ドア

P 名前付きパイプ (FIFO)

UNIX ファイルアクセス権

次の表に、各ユーザークラスに与えることができる、ファイルまたはディレクトリのアクセス権を示します。

表 6–2 ファイルとディレクトリのアクセス権

シンボル	アクセス権	オブジェクト	説明
r	読み込み	ファイル	ファイルの内容を開いて読み込むことができます。
		ディレクトリ	ディレクトリ内のファイルを一覧表示できます。
w	書き込み	ファイル	ファイルの内容の変更またはファイルの削除を行えます。
		ディレクトリ	ディレクトリに対してファイルまたはリンクを追加できます。また、ディレクトリ内のファイルまたはリンクの削除も行えます。
x	実行	ファイル	ファイルがプログラムまたはシェルスクリプトの場合、そのファイルを実行できます。また、exec(2) システム呼び出しの 1 つを使用してそのプログラムを実行することもできます。
		ディレクトリ	ディレクトリ内のファイルを開いたり、実行したりできます。また、ディレクトリを作成し、その下にサブディレクトリを作成できます。
-	拒否	ファイルとディレクトリ	ファイルの読み込み、書き込み、または実行を行うことができません。

これらのファイルアクセス権は、通常のファイルと特殊ファイル (デバイス、ソケット、名前付きパイプ (FIFO) など) の両方に適用されます。

シンボリックリンクには、そのリンクが指すファイルのアクセス権が適用されます。

ディレクトリとそのサブディレクトリ内のファイルは、そのディレクトリに対するファイルアクセス権を制限することで保護できます。ただし、スーパーユーザーはシステム上のすべてのファイルとディレクトリにアクセスできます。

特殊なファイルアクセス権 (setuid、 setgid、 スティッキービット)

実行可能ファイルと公開ディレクトリには、 3 種類の特殊なアクセス権 (setuid、 setgid、 およびスティッキービット) を設定できます。これらのアクセス権を設定すると、その実行可能ファイルを実行するユーザーは、そのファイルの所有者 (またはグループ) の ID を持つことができます。

特殊なアクセス権はセキュリティー上の問題を引き起こすため、設定するときは十分な注意が必要です。たとえば、ユーザー ID (UID) を 0 (root の UID) に設定するプログラムを実行すれば、ユーザーはスーパーユーザー権限を取得できます。また、すべてのユーザーは、所有するファイルに対して特殊なアクセス権を設定できるため、これもセキュリティー上の問題の原因となります。

setuid アクセス権や setgid アクセス権を不正に使用してスーパーユーザー権限が取得されていないか、システムを監視する必要があります。疑わしいアクセス権によって、管理プログラムの所有権が root や bin ではなく一般ユーザーに付与されていることが考えられます。この特殊なアクセス権を使用しているファイルをすべて検索し、リストする方法は、155 ページの「特殊なファイルアクセス権が設定されたファイルを見つける方法」を参照してください。

setuid アクセス権

setuid アクセス権を実行可能ファイルに設定すると、このファイルを実行するプロセスにはその実行可能ファイルを実行しているユーザーではなく、ファイルの所有者に基づいてアクセス権が与えられます。この特殊なアクセス権を使用すると、通常は所有者しか利用できないファイルやディレクトリにアクセスできます。

たとえば、passwd コマンドの setuid アクセス権によってユーザーはパスワードを変更できます。次に、setuid アクセス権のある passwd コマンドの例を示します。

```
-r-sr-sr-x   3 root     sys       28144 Jun 17 12:02 /usr/bin/passwd
```

この特殊なアクセス権は、プロセスの実行が終了したあとでも、高度な知識のあるユーザーは setuid プロセスによって与えられたアクセス権を維持する手段を見つけることができるため、セキュリティー上の危険が存在します。

注 – プログラムから予約済み UID (0 から 99) で setuid アクセス権を使用しても、実効 UID は正しく設定されない場合があります。シェルスクリプトを使用するか、setuid アクセス権では予約済み UID を使用しないようにしてください。

---

setgid アクセス

setgidアクセス権は setuid アクセス権に似ています。プロセスの実効グループ ID (GID) はファイルを所有するグループに変更され、ユーザーにはそのグループに与え

られたアクセス権に基づくアクセス権が与えられます。/usr/bin/mail コマンドには、次のように setgid アクセス権が設定されています。

```
-r-x--s--x   1 root    mail     67504 Jun 17 12:01 /usr/bin/mail
```

setgidアクセス権がディレクトリに適用されると、このディレクトリ内で作成されたファイルはディレクトリが所属するグループに含まれることになります。生成するプロセスが所属するグループに含まれるわけではありません。ディレクトリに対する書き込み権および実行権を持つユーザーは、そのディレクトリにファイルを作成できます。ただし、作成したファイルはユーザーのグループではなくディレクトリを所有するグループに割り当てられます。

setgidアクセス権を使用して不正にスーパーユーザー権限が取得されていないか、システムを監視する必要があります。疑わしいアクセス権によって、このようなプログラムへのグループアクセス権が、root や bin ではなく、意外なグループに与えられることがあります。このようなアクセス権を使用しているファイルをすべて検索し、一覧表示する方法は、155 ページの「特殊なファイルアクセス権が設定されたファイルを見つける方法」を参照してください。

スティッキービット

「スティッキービット」は、ディレクトリ内のファイルを保護するアクセス権ビットです。ディレクトリにスティッキービットが設定されている場合、そのファイルを削除できるのはその所有者、ディレクトリの所有者、または特権を持つユーザーだけです。特権を持つユーザーの例として、root ユーザーや Primary Administrator 役割が挙げられます。スティッキービットにより、ユーザーは /tmp などの公開ディレクトリからほかのユーザーのファイルを削除できなくなります。

```
drwxrwxrwt 7  root  sys   400 Sep  3 13:37 tmp
```

TMPFS ファイルシステム上で公開ディレクトリを設定するときには、スティッキービットを手動で設定してください。手順については、例 6–5 を参照してください。

umask のデフォルト値

ファイルやディレクトリを作成するときには、デフォルトのアクセス権が設定されます。このシステムデフォルトは変更が可能です。テキストファイルのアクセス権は 666 であり、すべてのユーザーに読み取り権と書き込み権を与えます。ディレクトリと実行可能ファイルのアクセス権は 777 で、すべてのユーザーに読み取り権、書き込み権、および実行権を与えます。一般にユーザーは、自分の /etc/profile ファイル、.cshrc ファイル、または .login ファイルを手動で指定し、システムのデフォルトに優先させます。

umask コマンドによって割り当てられる値は、デフォルトから差し引かれます。この処理には、chmod コマンドでアクセス権を与えるのと同じ方法でアクセス権を拒否する効果があります。たとえば、chmod 022 コマンドはグループとその他のユーザーに書き込み権を与えますが、umask 022 コマンドはグループとその他のユーザーの書き込み権を拒否します。

次の表に、代表的な umask の設定とその設定が実行可能ファイルに与える影響を示します。

表 6–3　各セキュリティーレベルの umask 設定

セキュリティーレベル	umask 設定	許可されないアクセス権
緩やか (744)	022	グループとその他のユーザーによる w
中程度 (740)	027	グループによる w、その他のユーザーによる rwx
中程度 (741)	026	グループによる w、その他のユーザーによる rw
厳しい (700)	077	グループとその他のユーザーによる rwx

umask 値の設定の詳細については、umask(1) のマニュアルページを参照してください。

ファイルアクセス権を設定するモード

chmod コマンドを使用すると、ファイルのアクセス権を変更できます。ファイルまたはディレクトリの所有者、あるいはスーパーユーザーだけがそのアクセス権を変更できます。

chmod コマンドを使用して、次のどちらかのモードでアクセス権を設定できます。

- 絶対モード – ファイルアクセス権を表す数値を使用します。絶対モードを使用してアクセス権を変更するときは、3 つ 1 組のアクセス権を 8 進数で表します。絶対モードは、アクセス権の設定に一番多く使用されている方法です。
- 記号モード – 英字と記号の組み合わせを使用して、アクセス権を追加または削除します。

次の表に、絶対モードでファイルのアクセス権を設定するための 8 進数値を示します。これらの数字を 3 つ組み合わせて、所有者、グループ、その他のユーザーのファイルアクセス権をこの順に設定します。たとえば、値 644 は、所有者に対して読み取り権と書き込み権を設定し、グループとその他のユーザーに対しては読み取り専用権を設定します。

表 6–4 絶対モードによるファイルのアクセス権の設定

8進数値	ファイルのアクセス権	設定されるアクセス権
0	---	なし
1	--x	実行権のみ
2	-w-	書き込み権のみ
3	-wx	書き込み権と実行権
4	r--	読み取り権のみ
5	r-x	読み取り権と実行権
6	rw-	読み取り権と書き込み権
7	rwx	読み取り権、書き込み権、実行権

次の表に、記号モードでファイルのアクセス権を設定するための記号の一覧を示します。記号では、アクセス権を設定または変更できる対象ユーザー、実行される操作、あるいは割り当てるまたは変更するアクセス権を指定できます。

表 6–5 記号モードによるファイルのアクセス権の設定

シンボル	機能	説明
u	<対象ユーザー>	ユーザー (所有者)
g	<対象ユーザー>	グループ
o	<対象ユーザー>	その他のユーザー
a	<対象ユーザー>	すべて
=	オペレータ	割り当て
+	オペレータ	追加
-	オペレータ	削除
r	アクセス権 (アクセス権ビット)	読み込み
w	アクセス権 (アクセス権ビット)	書き込み
x	アクセス権 (アクセス権ビット)	実行
l	アクセス権 (アクセス権ビット)	強制ロック、setgid ビットはオン、グループ実行ビットはオフ

表6–5 記号モードによるファイルのアクセス権の設定 (続き)

シンボル	機能	説明
s	アクセス権 (アクセス権ビット)	setuid または setgid ビットはオン
t	アクセス権 (アクセス権ビット)	スティッキービットはオン、その他の実行ビットはオン

機能列に <対象ユーザー> <操作> <アクセス権> の順で、ファイルまたはディレクトリのアクセス権を変更する記号を指定します。

who アクセス権を変更する対象となるユーザーを指定します。

operator 実行する操作を指定します。

permissions 変更するアクセス権を指定します。

絶対モードまたは記号モードで、ファイルに特殊なアクセス権を設定できます。しかし、ディレクトリに setuid アクセス権を設定する場合と、ディレクトリからこのアクセス権を削除する場合は、記号モードを使用する必要があります。絶対モードでは、3つ1組のアクセス権の左端に新しい8進数値を追加して、特殊なアクセス権を設定します。次の表に、ファイルに特殊なアクセス権を設定する8進数値を示します。

表6–6 絶対モードによる特殊なファイルアクセス権の設定

8進数値	特殊なファイルアクセス権
1	スティッキービット
2	setgid
4	setuid

アクセス制御リストによるUFSファイルの保護

従来のUNIXファイル保護機能は、ファイルの所有者、ファイルグループ、その他のユーザーという3つのユーザークラスに読み取り権、書き込み権、実行権を提供します。UFSファイルシステムでは、アクセス制御リスト (ACL) により次のことが可能となり、ファイルセキュリティーを管理するレベルがさらに詳細になります。

- ファイル所有者、グループ、その他のユーザー、特定のユーザーおよびグループにファイルアクセス権を定義します
- これらの各カテゴリにデフォルトのアクセス権を定義します

注 – ZFS ファイルシステムの ACL および NFSv4 ファイルの ACL については、『Oracle Solaris ZFS 管理ガイド』の第 8 章「ACL および属性を使用した Oracle Solaris ZFS ファイルの保護」を参照してください。

たとえば、グループ内のすべてのユーザーがファイルを読み取れるようにする場合は、そのファイルにグループの読み取り権を設定すればすみます。その場合に、そのグループ内の 1 人のユーザーだけに書き込み権を与えたいとします。標準の UNIX ではファイルセキュリティーをこのように設定することはできませんが、ACL では可能です。

UFS ファイルシステムでは、ACL エントリは `setfacl` コマンドを使ってファイルに設定されます。UFS ACL エントリは、次のようにコロンで区切ったフィールドで構成されます。

entry-type:[*uid*|*gid*]:*perms*

entry-type ファイルのアクセス権を設定する ACL エントリの種類です。たとえば、*entry-type* は `user` (ファイルの所有者) または `mask` (ACL マスク) に設定できます。ACL エントリの一覧については、表 6–7 と表 6–8 を参照してください。

uid ユーザー名またはユーザー ID (UID) です。

gid グループ名またはグループ ID (GID) です。

perms *entry-type* に設定するアクセス権を表します。*perms* は、記号文字 `rwx` または 8 進数の数字で指定できます。これらは `chmod` コマンドに使用するのと同じ数字です。

次に、ユーザー `stacey` の読み取り権と書き込み権を設定する ACL エントリの例を示します。

```
user:stacey:rw-
```

注意 – ACL などの UFS ファイルシステム属性は UFS ファイルシステムだけでサポートされます。そのため、`/tmp` ディレクトリ (通常は、TMPFS ファイルシステムとしてマウントされている) で ACL エントリを持つファイルを復元またはコピーすると、その ACL エントリは失われます。UFS ファイルを一時的に格納するには、`/var/tmp` ディレクトリを使用してください。

UFS ファイルの ACL エントリ

次の表は、ファイルに ACL を設定するときに使用する有効な ACL エントリの一覧です。最初の 3 つの ACL エントリは、基本的な UNIX のファイル保護機能を提供します。

表 6–7 UFS ファイルの ACL エントリ

ACL エントリ	説明
`u[ser]::`*perms*	所有者のアクセス権。
`g[roup]::`*perms*	グループのアクセス権。
`o[ther]:`*perms*	所有者やグループのメンバー以外のユーザーのアクセス権。
`m[ask]:`*perms*	ACL マスク。マスクエントリは、ユーザー (所有者以外) とグループに許可される最大アクセス権を示します。マスクは、すべてのユーザーとグループのアクセス権を即時に変更する手段です。 たとえば、`mask:r--` マスクエントリは、ユーザーとグループが書き込み権および実行権を持つことがアカウントに示されていても、読み取り権しか使用できないことを意味します。
`u[ser]:`*uid:perms*	特定のユーザーのアクセス権。*uid* には、ユーザー名または UID の数値を指定できます。
`g[roup]:`*gid:perms*	特定のグループのアクセス権。*gid* には、グループ名または GID の数値を指定できます。

UFS ディレクトリの ACL エントリ

表 6–7 に示した ACL エントリのほかに、ディレクトリにはデフォルトの ACL エントリも設定できます。デフォルトの ACL エントリを持つディレクトリ内で作成されたファイルまたはディレクトリは、デフォルトの ACL エントリと同じ ACL エントリを持つことになります。表 6–8 は、ディレクトリに使用するデフォルトの ACL エントリの一覧です。

ディレクトリ上の特定のユーザーおよびグループに対してデフォルトの ACL エントリを初めて設定するときは、所有者、グループ、その他のユーザー、および ACL マスクにもデフォルトの ACL エントリを設定する必要があります。これらのエントリは、必ず設定しなければなりません。これらは、次の表に示された最初の 4 つのデフォルト ACL エントリに相当します。

表 6–8 UFS ディレクトリのデフォルト ACL エントリ

デフォルトの ACL エントリ	説明
`d[efault]:u[ser]::`*perms*	所有者のデフォルトアクセス権。

表 6–8 UFS ディレクトリのデフォルト ACL エントリ (続き)

デフォルトの ACL エントリ	説明
d[efault]:g[roup]::*perms*	グループのデフォルトアクセス権。
d[efault]:o[ther]:*perms*	所有者やグループのメンバー以外のユーザーのデフォルトアクセス権。
d[efault]:m[ask]:*perms*	デフォルトの ACL マスク。
d[efault]:u[ser]:*uid:perms*	特定のユーザーのデフォルトアクセス権。*uid* には、ユーザー名または UID の数値を指定できます。
d[efault]:g[roup]:*gid:perms*	特定のグループのデフォルトアクセス権。*gid* には、グループ名または GID の数値を指定できます。

UFS ACL を制御するコマンド

次のコマンドは、UFS ファイルまたはディレクトリに設定される ACL の管理に使用されます。

setfacl コマンド　ACL エントリの設定、追加、変更、および削除を行います。詳細は、setfacl(1) のマニュアルページを参照してください。

getfacl コマンド　ACL エントリを表示します。詳細は、getfacl(1) のマニュアルページを参照してください。

実行可能ファイルを原因とするセキュリティーへの悪影響を防止する

セキュリティーのバグの多くは、デフォルトで読み取り権、書き込み権、および実行権が設定された実行可能スタックで発生します。実行可能スタックには実行権が割り当てられていますが、ほとんどのプログラムは実行可能スタックがなくても正しく機能します。

noexec_user_stack 変数を使うと、スタックを実行可能として割り当てるかどうかを指定できます。この変数は、Solaris 2.6 から利用可能です。デフォルトでは、この変数は 0 に設定されるため (64 ビットアプリケーションを除く)、プログラムは ABI に準拠して動作します。この変数が 0 以外の値に設定された場合、システムはシステム中のすべてのプロセスのスタックに読み取り権と書き込み権のマークを付けますが、実行権のマークは付けません。

この変数が設定されている場合、プログラムがスタック上でコードを実行しようとすると SIGSEGV シグナルが送信されます。このシグナルが送信されると、通常、プログラムはコアダンプして終了します。このようなプログラムは、違反しているプロ

グラム名、プロセス ID、 およびプログラムを実行した実ユーザー ID を含む警告メッセージも生成します。次に例を示します。

```
a.out[347] attempt to execute code on stack by uid 555
```

メッセージは、syslog kern 機能が notice レベルの設定されているときに、syslog デーモンによってログに記録されます。このログへの記録は、デフォルトで syslog.conf ファイルに設定されていて、メッセージがコンソールと /var/adm/messages ファイルの両方に送信されることを意味します。詳細は、syslogd(1M) および syslog.conf(4) のマニュアルページを参照してください。

syslog メッセージは、潜在的なセキュリティーの問題を調べるときに役立ちます。また、このメッセージは、この変数を設定したために正しく動作しなくなった、実行可能スタックに依存するプログラムを確認するのにも役立ちます。メッセージを記録しない場合、システム管理者は、/etc/system ファイルで noexec_user_stack_log 変数を 0 に設定します。メッセージが記録されなくなりますが、SIGSEGV シグナルは送られるので、実行中のプログラムはコアダンプを生成して終了します。

mprotect() 関数を使用すれば、プログラムのスタックが実行可能であることを明示的にマーク付けできます。詳細は、mprotect(2) のマニュアルページを参照してください。

ハードウェアの制限のため、実行可能スタックの問題を捕えて報告する機能は、ほとんどの x86 ベースシステムで利用できません。AMD64 プロダクトファミリ内のシステムであれば、実行可能スタックの問題を捕えて報告できます。

ファイルの保護 (作業マップ)

次の作業マップは、ファイル保護に関連した作業の説明箇所を示しています。

作業	説明	参照先
UNIX アクセス権を使用してファイルを保護します	ファイルの UNIX アクセス権を表示します。UNIX アクセス権を使用してファイルを保護します	143 ページの「UNIX アクセス権によるファイルの保護 (作業マップ)」
ACL を使用してファイルを保護します	ACL を追加することで、UNIX アクセス権よりも細かいレベルでファイルを保護します。	149 ページの「ACL による UFS ファイルの保護 (作業マップ)」
セキュリティーリスクのあるファイルからシステムを保護します	不審な所有権が設定された実行可能ファイルを見つけます。システムに損傷を与えかねないファイルを無効にします。	154 ページの「セキュリティーリスクのあるプログラムからの保護 (作業マップ)」

UNIX アクセス権によるファイルの保護 (作業マップ)

次の作業マップは、ファイルアクセス権の一覧表示、ファイルアクセス権の変更、特殊なファイルアクセス権によるファイルの保護などの作業操作について説明した箇所を示しています。

作業	参照先
ファイル情報を表示します	143 ページの「ファイル情報を表示する方法」
ファイル所有権を変更します	144 ページの「ローカルファイルの所有者を変更する方法」 145 ページの「ファイルのグループ所有権を変更する方法」
ファイルアクセス権を変更します	146 ページの「ファイルアクセス権を記号モードで変更する方法」 146 ページの「ファイルアクセス権を絶対モードで変更する方法」 148 ページの「特殊なファイルアクセス権を絶対モードで変更する方法」

▼ ファイル情報を表示する方法

`ls` コマンドを使用して、ディレクトリ内のすべてのファイルに関する情報を表示します。

- 次のコマンドを入力すると、現在のディレクトリ内のすべてのファイルの一覧が長形式で表示されます。

  ```
  % ls -la
  ```

  `-l` ユーザー所有権、グループ所有権、ファイルのアクセス権などを長形式で表示します。

  `-a` ドット (.) で始まる隠しファイルを含め、すべてのファイルを表示します。

例 6–1 ファイル情報を表示する

次の例では、`/sbin` ディレクトリ内のファイルを部分的に表示しています。

```
% cd /sbin
% ls -la
total 13456
drwxr-xr-x   2 root     sys          512 Sep  1 14:11 .
drwxr-xr-x  29 root     root        1024 Sep  1 15:40 ..
-r-xr-xr-x   1 root     bin       218188 Aug 18 15:17 autopush
lrwxrwxrwx   1 root     root          21 Sep  1 14:11 bpgetfile -> ...
-r-xr-xr-x   1 root     bin       505556 Aug 20 13:24 dhcpagent
-r-xr-xr-x   1 root     bin       456064 Aug 20 13:25 dhcpinfo
-r-xr-xr-x   1 root     bin       272360 Aug 18 15:19 fdisk
-r-xr-xr-x   1 root     bin       824728 Aug 20 13:29 hostconfig
```

```
-r-xr-xr-x   1 root     bin       603528 Aug 20 13:21 ifconfig
-r-xr-xr-x   1 root     sys       556008 Aug 20 13:21 init
-r-xr-xr-x   2 root     root      274020 Aug 18 15:28 jsh
-r-xr-xr-x   1 root     bin       238736 Aug 21 19:46 mount
-r-xr-xr-x   1 root     sys         7696 Aug 18 15:20 mountall
   .
   .
   .
```

それぞれの行には、ファイルについての情報が次の順で表示されています。

- ファイルの形式 – たとえば、d。ファイル形式の一覧は、132 ページの「ファイルとディレクトリの所有権」を参照してください。
- アクセス権 – たとえば、r-xr-xr-x。詳細は、132 ページの「ファイルとディレクトリの所有権」を参照してください。
- ハードリンクの数 – たとえば、2。
- ファイルの所有者 – たとえば、root。
- ファイルのグループ – たとえば、bin。
- バイト数で示したファイルサイズ – たとえば、7696。
- ファイルの作成日時、またはファイルが最後に変更された日時 – たとえば、Aug 18 15:20。
- ファイルの名前 – たとえば、mountall。

▼ ローカルファイルの所有者を変更する方法

ファイル所有者、Primary Administrator 役割、またはスーパーユーザーは、任意のファイルの所有権を変更できます。

1 ファイルのアクセス権を表示します。

```
% ls -l example-file
-rw-r--r--   1 janedoe   staff   112640 May 24 10:49 example-file
```

2 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

3 ファイルの所有者を変更します。

```
# chown stacey example-file
```

4 ファイルの所有者が変更されていることを確認します。

```
# ls -l example-file
-rw-r--r--   1 stacey   staff   112640 May 26 08:50 example-file
```

例 6–2 自分のファイルの所有権を変更するためのユーザーの有効化

セキュリティー上の考慮事項 – rstchown 変数の設定をゼロに変更するにはそれ相応の理由が必要です。この設定により、ユーザーはファイルの所有権を別のユーザー名に変更できます。

この例では、rstchown 変数の値は、/etc/system ファイル内でゼロに設定されます。この設定によりファイルの所有者は、chown コマンドを使用してファイルの所有権をほかのユーザーに変更できます。所有者は chgrp コマンドを使用し、ファイルのグループ所有権を所有者自身が属していないグループに設定することもできます。変更は、システムの再起動時に適用されます。

```
set rstchown = 0
```

詳細は、chown(1) および chgrp(1) のマニュアルページを参照してください。

NFS マウントされたファイルシステムでは、所有権とグループの変更に関する制限がほかにもあります。NFS マウントされたシステムに対するアクセスの制限については、『Solaris のシステム管理 (ネットワークサービス)』の第 6 章「ネットワークファイルシステムへのアクセス (リファレンス)」を参照してください。

▼ ファイルのグループ所有権を変更する方法

1. **Primary Administrator 役割を引き受けるか、スーパーユーザーになります。**

   Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2. ファイルのグループ所有権を変更します。

   ```
   $ chgrp scifi example-file
   ```

   グループの設定については、『Solaris のシステム管理 (基本編)』の第 4 章「ユーザーアカウントとグループの管理 (概要)」を参照してください。

3. ファイルのグループ所有権が変更されていることを確認します。

   ```
   $ ls -l example-file
    -rw-r--r--   1 stacey   scifi   112640 June 20 08:55  example-file
   ```

   また、例 6–2 も参照してください。

▼ ファイルアクセス権を記号モードで変更する方法

1 ファイルまたはディレクトリの所有者でない場合は、スーパーユーザーになるか、同等の役割を引き受けます。

現在の所有者またはスーパーユーザーだけが、chmod コマンドを使用してファイルまたはディレクトリの所有者を変更できます。

2 アクセス権を記号モードで変更します。

```
% chmod who operator permissions filename
```

who アクセス権を変更する対象となるユーザーを指定します。

operator 実行する操作を指定します。

permissions 変更するアクセス権を指定します。有効な記号の一覧は、表 6–5 を参照してください。

filename ファイルまたはディレクトリを指定します。

3 ファイルのアクセス権が変更されていることを確認します。

```
% ls -l filename
```

例 6–3 アクセス権を記号モードで変更する

次の例では、その他のユーザーから読み取り権を削除します。

```
% chmod o-r example-file1
```

次の例では、ユーザー、グループ、およびその他のユーザーに、読み取り権と実行権を追加します。

```
$ chmod a+rx example-file2
```

次の例では、グループに読み取り権、書き込み権、および実行権を割り当てます。

```
$ chmod g=rwx example-file3
```

▼ ファイルアクセス権を絶対モードで変更する方法

1 ファイルまたはディレクトリの所有者でない場合は、スーパーユーザーになるか、同等の役割を引き受けます。

現在の所有者またはスーパーユーザーだけが、chmod コマンドを使用してファイルまたはディレクトリの所有者を変更できます。

2 アクセス権を絶対モードで変更します。

```
% chmod nnn filename
```

nnn 所有者、グループ、その他のユーザーのアクセス権をこの順序で表す 8 進数値を指定します。有効な 8 進数値の一覧は、表 6–4 を参照してください。

filename ファイルまたはディレクトリを指定します。

注 – chmod コマンドを使用して ACL エントリを持つファイルのグループアクセス権を変更する場合、グループアクセス権と ACL マスクの両方が新しいアクセス権に変更されます。新しい ACL マスクのアクセス権は、そのファイル上に ACL エントリを持つほかのユーザーおよびグループのアクセス権を変更する場合があるので注意が必要です。getfacl コマンドを使用して、すべての ACL エントリに適切なアクセス権が設定されていることを確認してください。詳細は、getfacl(1) のマニュアルページを参照してください。

---

3 ファイルのアクセス権が変更されていることを確認します。

```
% ls -l filename
```

例 6–4 アクセス権を絶対モードで変更する

次の例では、公開ディレクトリのアクセス権が、744 (読み取り / 書き込み / 実行; 読み取り専用; 読み取り専用) から 755 (読み取り / 書き込み / 実行; 読み取り / 実行; 読み取り / 実行) に変更されます。

```
# ls -ld public_dir
drwxr--r--  1 jdoe   staff    6023 Aug  5 12:06 public_dir
# chmod 755 public_dir
# ls -ld public_dir
drwxr-xr-x  1 jdoe   staff    6023 Aug  5 12:06 public_dir
```

次の例では、実行可能シェルスクリプトのアクセス権が、読み取り/書き込みから、読み取り/書き込み/実行に変更されます。

```
% ls -l my_script
-rw------- 1 jdoe   staff    6023 Aug  5 12:06 my_script
% chmod 700 my_script
% ls -l my_script
-rwx------ 1 jdoe   staff    6023 Aug  5 12:06 my_script
```

▼ 特殊なファイルアクセス権を絶対モードで変更する方法

1. ファイルまたはディレクトリの所有者でない場合は、スーパーユーザーになるか、同等の役割を引き受けます。

   現在の所有者またはスーパーユーザー能力を持つユーザーだけが、chmod コマンドを使用してファイルまたはディレクトリの所有者を変更できます。

2. 特殊なアクセス権を絶対モードで変更します。

   ```
   % chmod nnnn filename
   ```

   nnnn ファイルまたはディレクトリのアクセス権を変更する 8 進数値を指定します。一番左端の 8 進数値で、ファイルに特殊なアクセス権を設定します。特殊なアクセス権に有効な 8 進数値の一覧は、表 6–6 を参照してください。

   filename ファイルまたはディレクトリを指定します。

   注 – chmod コマンドを使用して ACL エントリを持つファイルのグループアクセス権を変更する場合、グループアクセス権と ACL マスクの両方が新しいアクセス権に変更されます。新しい ACL マスクのアクセス権は、そのファイル上に ACL エントリを持つ追加ユーザーおよびグループのアクセス権を変更する場合があるので注意が必要です。getfacl コマンドを使用して、すべての ACL エントリに適切なアクセス権が設定されていることを確認してください。詳細は、getfacl(1) のマニュアルページを参照してください。

   ---

3. ファイルのアクセス権が変更されていることを確認します。

   ```
   % ls -l filename
   ```

例 6–5 絶対モードによる特殊なファイルアクセス権の設定

次の例は、dbprog ファイルに setuid のアクセス権を設定します。

```
# chmod 4555 dbprog
# ls -l dbprog
-r-sr-xr-x   1 db     staff        12095 May  6 09:29 dbprog
```

次の例では、dbprog2 ファイルに setgid のアクセス権を設定します。

```
# chmod 2551 dbprog2
# ls -l dbprog2
-r-xr-s--x   1 db     staff       24576 May  6 09:30 dbprog2
```

次の例では、public_dir ディレクトリにスティッキービットのアクセス権を設定します。

```
# chmod 1777 public_dir
# ls -ld public_dir
drwxrwxrwt   2 jdoe   staff          512 May 15 15:27 public_dir
```

ACL による UFS ファイルの保護 (作業マップ)

次の作業マップは、UFS ファイルの ACL を表示する、ACL を変更する、ほかのファイルへ ACL をコピーするといった作業について説明した箇所を示しています。

作業	参照先
ファイルに ACL が存在するかを確認します	149 ページの「ファイルに ACL が設定されているかどうかを検査する方法」
ファイルに ACL を追加します	150 ページの「ファイルに ACL エントリを追加する方法」
ACL をコピーします	151 ページの「ACL をコピーする方法」
ACL を変更します	152 ページの「ファイルの ACL エントリを変更する方法」
ファイルから ACL を削除します	153 ページの「ファイルから ACL エントリを削除する方法」
ファイルの ACL を表示します	153 ページの「ファイルの ACL エントリを表示する方法」

▼ ファイルに ACL が設定されているかどうかを検査する方法

- ファイルに **ACL** が設定されているかをチェックします。

  ```
  % ls -l filename
  ```

  filename には、ファイルまたはディレクトリを指定します。

  出力のモードフィールドの右側にプラス記号 (+) が表示されているときは、そのファイルに ACL が設定されています。

注 – UNIX ファイルアクセス権を超える ACL エントリを追加しない限り、ファイルの ACL は「弱い」とみなされ、プラス記号 (+) は表示されません。

例 6–6 ファイルに ACL が設定されているかどうかを検査する

次の例では、ch1.sgm ファイルに ACL が設定されています。ACL は、モードフィールドの右側にあるプラス記号 (+) で示されています。

```
% ls -l ch1.sgm
-rwxr-----+  1 stacey   techpubs      167 Nov 11 11:13 ch1.sgm
```

▼ ファイルに ACL エントリを追加する方法

1 **setfacl コマンドを使用してファイルの ACL を設定します。**

```
% setfacl -s user::perms,group::perms,other:perms,mask:perms,acl-entry-list  filename ...
```

`-s`	ファイルに対して ACL を設定します。すでに ACL が設定されている場合、新しい ACL に置き換えます。このオプションには、少なくとも `user::`、 `group::`、 および `other::` のエントリを指定する必要があります。
`user::`*perms*	所有者のアクセス権を指定します。
`group::`*perms*	グループメンバーのアクセス権を指定します。
`other:`*perms*	所有者またはグループのメンバー以外のユーザーのアクセス権を指定します。
`mask:`*perms*	ACL マスクのアクセス権を指定します。マスクは、ユーザー (所有者以外) とグループに許される最大アクセス権を示します。
acl-entry-list	ファイルまたはディレクトリ上で特定のユーザーとグループに設定する 1 つまたは複数の ACL エントリのリストを指定します。ディレクトリ上でデフォルトの ACL エントリを設定することもできます。有効な ACL エントリについては、表 6–7 と表 6–8 を参照してください。
filename ...	ACL を設定する 1 つまたは複数のファイルまたはディレクトリを指定します。*filename* が複数ある場合は、スペースで区切ります。

注意 – すでにファイル上に ACL が存在する場合、`-s` オプションを指定すると、ACL 全体が新しい ACL に置き換えられます。

詳細は、`setfacl(1)` のマニュアルページを参照してください。

2 **ACL (ACL エントリ) がファイルに設定されたことを確認します。**

```
% getfacl filename
```

詳細は、149 ページの「ファイルに ACL が設定されているかどうかを検査する方法」を参照してください。

例 6–7 ファイルの ACL を設定する

次の例では、ch1.sgm ファイルのアクセス権を設定しています。所有者には読み取り権と書き込み権が設定され、グループには読み取り専用権が設定され、その他のユーザーには何も設定されません。また、ユーザー anusha にはこのファイルの読み取り権および書き込み権が与えられ、ACL マスクには読み取り権および書き込み権が設定されます。これは、ユーザーやグループは実行権を持たないことを意味します。

```
% setfacl -s user::rw-,group::r--,other:---,mask:rw-,user:anusha:rw- ch1.sgm
% ls -l
total 124
-rw-r-----+  1 stacey  techpubs   34816 Nov 11 14:16 ch1.sgm
-rw-r--r--   1 stacey  techpubs   20167 Nov 11 14:16 ch2.sgm
-rw-r--r--   1 stacey  techpubs    8192 Nov 11 14:16 notes
% getfacl ch1.sgm
# file: ch1.sgm
# owner: stacey
# group: techpubs
user::rw-
user:anusha:rw-    #effective:rw-
group::r--         #effective:r--
mask:rw-
other:---
```

次の例では、ch2.sgm ファイルのアクセス権を設定します。所有者には読み取り権、書き込み権、および実行権が設定され、グループには読み取り専用権が設定され、その他のユーザーには何も設定されません。また、ACL マスクには読み取り権が設定されます。さらに、ユーザー anusha には読み取り権と書き込み権が与えられます。ただし、ACL マスクの設定により、anusha のアクセス権は読み取り専用です。

```
% setfacl -s u::7,g::4,o:0,m:4,u:anusha:7 ch2.sgm
% getfacl ch2.sgm
# file: ch2.sgm
# owner: stacey
# group: techpubs
user::rwx
user:anusha:rwx         #effective:r--
group::r--              #effective:r--
mask:r--
other:---
```

▼ ACL をコピーする方法

- **getfacl** の出力先を変更することにより、ファイルの **ACL** をほかのファイルへコピーします。

  ```
  % getfacl filename1 | setfacl -f - filename2
  ```

filename1 ACL のコピー元ファイルを指定します

filename2 ACL のコピー先ファイルを指定します

例 6–8 ACL をコピーする

次の例では、ch2.sgm の ACL が ch3.sgm にコピーされます。

```
% getfacl ch2.sgm | setfacl -f - ch3.sgm
```

▼ ファイルの ACL エントリを変更する方法

1 **setfacl** コマンドを使用してファイルの **ACL** エントリを変更します。

```
% setfacl -m acl-entry-list filename ...
```

`-m` 既存の ACL エントリを変更します。

acl-entry-list ファイルまたはディレクトリで変更する 1 つまたは複数の ACL エントリのリストを指定します。ディレクトリのデフォルト ACL エントリを変更することもできます。有効な ACL エントリについては、表 6–7 と表 6–8 を参照してください。

filename ... 1 つまたは複数のファイルまたはディレクトリを空白で区切って指定します。

2 ファイルの **ACL** エントリが変更されたことを確認します。

```
% getfacl filename
```

例 6–9 ファイルの ACL エントリを変更する

次の例では、ユーザー anusha のアクセス権を読み取りおよび書き込みに変更します。

```
% setfacl -m user:anusha:6 ch3.sgm
% getfacl ch3.sgm
# file: ch3.sgm
# owner: stacey
# group: techpubs
user::rw-
user::anusha:rw-        #effective:r--
group::r-               #effective:r--
mask:r--
other:r-
```

次の例では、book ディレクトリのデフォルトアクセス権を変更します。グループ staff のデフォルトのアクセス権を読み取りに変更し、さらに、ACL マスクのデフォルトアクセス権を読み取りおよび書き込みに変更します。

```
% setfacl -m default:group:staff:4,default:mask:6 book
```

▼ ファイルから ACL エントリを削除する方法

1 ファイルから **ACL** エントリを削除します。

```
% setfacl -d acl-entry-list filename ...
```

`-d` 指定した ACL エントリを削除します。

acl-entry-list ファイルまたはディレクトリから (アクセス権を指定せずに) 削除する ACL エントリのリストを指定します。特定のユーザーとグループの ACL エントリとデフォルトの ACL エントリ以外は削除できません。有効な ACL エントリについては、表 6–7 と表 6–8 を参照してください。

filename ... 1 つまたは複数のファイルまたはディレクトリを空白で区切って指定します。

`setfacl -s` コマンドを使用してファイルのすべての ACL エントリを削除してから、指定した新しい ACL エントリで置き換えることもできます。

2 ファイルから **ACL** エントリが削除されたことを確認します。

```
% getfacl filename
```

例 6–10 ファイルから ACL エントリを削除する

次の例では、`ch4.sgm` ファイルからユーザー `anusha` を削除します。

```
% setfacl -d user:anusha ch4.sgm
```

▼ ファイルの ACL エントリを表示する方法

- **`getfacl`** コマンドを使用してファイルの **ACL** エントリを表示します。

```
% getfacl [-a | -d] filename ...
```

`-a` 指定したファイルまたはディレクトリのファイル名、所有者、グループ、ACL エントリを表示します。

`-d` 指定したディレクトリのファイル名、所有者、グループ、デフォルトの ACL エントリ (存在する場合) を表示します。

filename ... 1 つまたは複数のファイルまたはディレクトリを空白で区切って指定します。

複数のファイル名をコマンド行に指定した場合は、各 ACL エントリの間に空白行が表示されます。

例 6–11 ファイルの ACL エントリを表示する

次の例は、ch1.sgm ファイルのすべての ACL エントリを示します。ユーザーエントリとグループエントリの隣りの #effective: は、ACL マスクによって変更されたあとのアクセス権の設定を示します。

```
% getfacl ch1.sgm

# file: ch1.sgm
# owner: stacey
# group: techpubs
user::rw-
user:anusha:r-        #effective:r--
group::rw-            #effective:rw-
mask:rw-
other:---
```

次の例は、book ディレクトリのデフォルトの ACL エントリを示します。

```
% getfacl -d book

# file: book
# owner: stacey
# group: techpubs
user::rwx
user:anusha:r-x       #effective:r-x
group::rwx            #effective:rwx
mask:rwx
other:---
default:user::rw-
default:user:anusha:r--
default:group::rw-
default:mask:rw-
default:other:---
```

セキュリティーリスクのあるプログラムからの保護 (作業マップ)

次の作業マップは、リスクのある実行ファイルをシステム内に見つける作業や、プログラムが実行可能スタックを不正利用しないように防ぐ作業などの操作を説明した箇所を示しています。

作業	説明	参照先
特殊なアクセス権を持つファイルを見つけます	setuid ビットがセットされているが、root ユーザーによって所有されていないというファイルを見つけます。	155 ページの「特殊なファイルアクセス権が設定されたファイルを見つける方法」
実行可能スタックがオーバーフローしないように防ぎます	プログラムが実行可能スタックを不正に利用しないように防ぎます。	156 ページの「プログラムが実行可能スタックを使用できないようにする方法」
実行可能スタックのメッセージがログに記録されるのを防ぎます	実行可能スタックのメッセージのログ記録を無効にします。	例 6–13

▼ 特殊なファイルアクセス権が設定されたファイルを見つける方法

管理者はシステムを監視し、プログラム上で承認を得ることなく setuid アクセス権および setgid アクセス権が使用されていないかをチェックする必要があります。setuid アクセス権と setgid アクセス権を使用すると、通常のユーザーでもスーパーユーザー権限を取得できてしまいます。疑わしい実行可能ファイルによって、所有権が root または bin ではなく、通常のユーザーに与えられることがあります。

1 **Primary Administrator 役割を引き受けるか、スーパーユーザーになります。**

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 **find コマンドを使用して setuid アクセス権が設定されているファイルを検索します。**

```
# find directory -user root -perm -4000 -exec ls -ldb {} \; >/tmp/filename
```

find *directory*	指定したディレクトリから始めて、マウントされているすべてのパスを検査します。ディレクトリとしてルート (/)、sys、bin、または mail を指定できます。
-user root	root が所有するファイルを表示します。
-perm -4000	アクセス権が 4000 に設定されているファイルだけを表示します。
-exec ls -ldb	find コマンドの出力を ls -ldb 形式で表示します。
/tmp/*filename*	find コマンドの結果が書き込まれるファイルです。

3 **結果を /tmp/*filename* に出力します。**

```
# more /tmp/filename
```

setuid アクセス権の詳細については、134 ページの「setuid アクセス権」を参照してください。

例 6–12 setuid アクセス権が設定されているファイルを検索する

次の例の出力は、rar というグループのユーザーが /usr/bin/sh のコピーを作成し、そのアクセス権を root への setuid に設定したことを示しています。この結果、/usr/rar/bin/sh プログラムは root アクセス権で実行されます。

この出力は、/var/tmp/chkprm ディレクトリを /export/sysreports/ckprm ディレクトリに移動することによって、今後の参照用に保存されています。

```
# find / -user root -perm -4000 -exec ls -ldb {} \; > /var/tmp/ckprm
# cat /var/tmp/ckprm
-r-sr-xr-x 1 root bin 38836 Aug 10 16:16 /usr/bin/at
-r-sr-xr-x 1 root bin 19812 Aug 10 16:16 /usr/bin/crontab
---s--x--x 1 root sys 46040 Aug 10 15:18 /usr/bin/ct
-r-sr-xr-x 1 root sys 12092 Aug 11 01:29 /usr/lib/mv_dir
-r-sr-sr-x 1 root bin 33208 Aug 10 15:55 /usr/lib/lpadmin
-r-sr-sr-x 1 root bin 38696 Aug 10 15:55 /usr/lib/lpsched
---s--x--- 1 root rar 45376 Aug 18 15:11 /usr/rar/bin/sh
-r-sr-xr-x 1 root bin 12524 Aug 11 01:27 /usr/bin/df
-rwsr-xr-x 1 root sys 21780 Aug 11 01:27 /usr/bin/newgrp
-r-sr-sr-x 1 root sys 23000 Aug 11 01:27 /usr/bin/passwd
-r-sr-xr-x 1 root sys 23824 Aug 11 01:27 /usr/bin/su
# mv /var/tmp/ckprm /export/sysreports/ckprm
```

▼ プログラムが実行可能スタックを使用できないようにする方法

実行可能スタックのセキュリティーリスクについては、141 ページの「実行可能ファイルを原因とするセキュリティーへの悪影響を防止する」を参照してください。

1. **Primary Administrator 役割を引き受けるか、スーパーユーザーになります。**

   Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2. **/etc/system ファイルを編集し、次の行を追加します。**

   ```
   set noexec_user_stack=1
   ```

3. **システムを再起動します。**

   ```
   # init 6
   ```

例 6–13 実行可能スタックメッセージのログ記録を無効にする

この例では、実行可能スタックメッセージのログ記録が無効にされ、続いてシステムの再起動が行われます。

```
# cat /etc/system
set noexec_user_stack=1
set noexec_user_stack_log=0
# init 6
```

◆ ◆ ◆ 第 7 章

7 自動セキュリティー拡張ツールの使用 (手順)

この章では、自動セキュリティー拡張ツール (ASET) を使用して、システムファイルおよびディレクトリへのアクセスを監視または制限する方法について説明します。

この章で説明する手順は次のとおりです。

ASET よりも広範なツールとしては、Oracle Solaris Security Toolkit を使用してください。Oracle Solaris Security Toolkit は、Oracle Solaris システムの強化と最小化のためのフレームワークを提供します。このツールキットには、プロファイリングツール、レポートツール、および取り消し機能が含まれています。詳細は、55 ページの「Oracle Solaris Security Toolkit の使用」を参照してください。

自動セキュリティー拡張ツール (ASET)

Oracle Solaris OS には、ASET が組み込まれています。ASET を使用すると、ほかの場合には手動で実行する作業が自動的に実行され、システムのセキュリティーを監視して制御できます。

ASET セキュリティーパッケージには、システムのセキュリティーを制御して監視できるように、自動管理ツールが組み込まれています。ASET を実行するセキュリティーレベルには、低レベル、中レベル、または高レベルを指定できます。上のレベルほど、ASET のファイル制御機能が増え、ファイルアクセスは減少し、システムセキュリティーが厳しくなります。

ASET には 7 つのタスクがあり、各タスクがシステムファイルに対して特定の検査と調整を行います。ASET のタスクはファイルのアクセス権を厳しくし、重要なシステムファイルの内容にセキュリティー上の弱点がないかどうかを確認して、重要な領域を監視します。ASET では、ゲートウェイシステムとして機能するシステムに

ファイアウォールシステムの基本要件を適用し、ネットワークも保護できます。詳細は、163 ページの「ファイアウォールの設定」を参照してください。

ASET は、構成用のマスターファイルを使用します。マスターファイルやレポートなどの ASET ファイルは、/usr/aset ディレクトリにあります。これらのファイルは、サイトの特定の要件に合わせて変更できます。

各タスクは、検出されたセキュリティー上の弱点と、タスクがシステムファイルに対して行った変更を示すレポートを生成します。最上位のセキュリティーレベルで実行すると、ASET はシステムセキュリティー上のすべての弱点を変更しようとします。潜在的なセキュリティー問題を解決できない場合、ASET は問題の存在を報告します。

ASET セッションを起動するには、/usr/aset/aset コマンドを対話的に使用します。ASET を定期的に実行する場合は、crontab ファイルにエントリを指定します。

ASET のタスクはディスクをかなり使用するため、通常の動作の妨げになることがあります。システム性能への影響を最小限度に抑えるために、24 時間ごとまたは 48 時間ごとに深夜など、システムの稼働レベルがもっとも低いときに ASET を実行するようにスケジュールしてください。

ASET のセキュリティーレベル

ASET は、 低、中、高の 3 つのセキュリティーレベルのいずれかで動作するように設定できます。上のレベルほど、ASET のファイル制御機能が増え、ファイルアクセスが減少し、システムのセキュリティーが厳しくなります。これらの機能には、ユーザーによるファイルアクセスを制限せずにシステムセキュリティーを監視する最低レベルから、システムが完全にセキュリティー保護される最高レベルまで、アクセス権が段階的に厳しくなります。

次の表で、この 3 つのセキュリティーレベルの概要について説明します。

セキュリティーレベル	説明
低	システムファイルの属性が標準リリース値に設定されることが保証されます。ASET は複数の検査を実行し、セキュリティー上の潜在的な弱点を報告します。低レベルでは、ASET は動作せず、システムサービスは影響を受けません。
中	ほとんどの環境で十分にセキュリティーが制御されます。ASET はシステムファイルとパラメータの設定の一部を変更し、システムアクセスを制限し、セキュリティー上の攻撃によるリスクを減らします。ASET は、セキュリティー上の弱点と、アクセスを制限するために行った変更を報告します。中レベルでは、ASET はシステムサービスに影響しません。

セキュリティーレベル	説明
高	システムに高度なセキュリティーが適用されます。ASET は多数のシステムファイルとパラメータの設定を調整して、アクセス権を最小限度に抑えます。ほとんどのシステムアプリケーションとコマンドは、正常に機能します。ただし、高レベルでは、セキュリティーがほかのシステムの動作より優先されます。

注 – 管理者がセキュリティーレベルを下げない限り、ASET によってファイルのアクセス権が変更されてファイルの安全性が低くなるということはありません。管理者が意図的にシステムを ASET 実行前の設定に戻すという場合もあります。

ASET のタスク一覧

この節では、ASET のタスクについて説明します。管理者は、ASET の各タスクを理解しておく必要があります。ASET の目的と処理、ASET が影響を与えるシステムコンポーネントなどを理解することで、レポート内容を判断し、効果的に活用できます。

ASET のレポートファイルには、各 ASET タスクで検出された問題をできるだけ詳細に記述するメッセージが含まれています。これらのメッセージによって、問題を診断して解決できます。ただし、ASET を活用するには、システム管理とシステム構成要素を全般的に理解していることが前提となります。システム管理者になったばかりのユーザーは、ほかの Oracle Solaris システム管理マニュアルを参照することをお勧めします。また、関連するマニュアルページを参照して、ASET 管理の概要を把握するとよいでしょう。

`taskstat` ユーティリティーは、完了したタスクとまだ実行中のタスクを識別します。完了したタスクごとにレポートファイルが生成されます。`taskstat` ユーティリティーの詳細は、`taskstat(1M)` を参照してください。

システムファイルのアクセス権の調整

このタスクでは、システムファイルのアクセス権を指定したセキュリティーレベルに設定します。このタスクは、システムのインストール時に実行されます。以前に設定したレベルをあとから変更する場合は、このタスクを再度実行してください。低セキュリティーレベルでは、アクセス権は開放型の情報共有環境に適した値に設定されています。中セキュリティーレベルでは、アクセス権はほとんどの環境に十分なセキュリティーが適用される程度に厳しくなります。高セキュリティーレベルでは、アクセス権がもっとも厳しく制限されます。

このタスクによってシステムファイルのアクセス権やパラメータの設定に加えられた変更は、tune.rpt ファイル内にレポートされます。ASET がアクセス権を設定するときに調整するファイルの例は、175 ページの「調整ファイルの例」を参照してください。

システムファイルの確認

このタスクでは、システムファイルが検査され、各ファイルと、マスターファイル内のそれらの記述とが比較されます。マスターファイルは、ASET がこのタスクを実行するときに初めて作成されます。マスターファイルには、指定したセキュリティーレベルの checklist によって適用されるシステムファイル設定が含まれています。

ファイルが確認されるディレクトリのリストは、セキュリティーレベルごとに定義されます。デフォルトのリストを使用するか、レベルごとに異なるディレクトリを指定してリストを変更できます。

ファイルごとに次の条件が確認されます。

- 所有者とグループ
- アクセス権ビット
- サイズとチェックサム
- リンク数
- 最終変更時刻

ASET によって矛盾が見つかると、cklist.rpt ファイルに記録されます。このファイルには、システムファイルのサイズ、アクセス権、およびチェックサムの値について、マスターファイルと比較した結果が入っています。

ユーザーとグループの確認

このタスクでは、ユーザーアカウントとグループの整合性と完全性が確認されます。このタスクは、passwd ファイルと group ファイル内の定義を使用します。ローカルパスワードファイルと、NIS または NIS+ パスワードファイルが検査されます。NIS+ のパスワードファイルの問題はレポートされますが、解決はされません。

このタスクでは、次の違反が検査されます。

- 重複名または重複 ID
- 正確でない形式のエントリ
- パスワードが付いていないアカウント
- 無効なログインディレクトリ
- アカウント nobody
- 空のグループパスワード
- NIS サーバーまたは NIS+ サーバー上の /etc/passwd ファイル内のプラス記号 (+)

矛盾は、usrgrp.rpt ファイル内にレポートされます。

システム構成ファイルの確認

このタスクでは、ASET はあらゆるシステムテーブルを検査します。テーブルのほとんどは /etc ディレクトリに入っています。

次のシステム構成ファイルが検査されます。

- /etc/default/login
- /etc/hosts.equiv
- /etc/inetd.conf
- /etc/aliases
- /var/adm/utmpx
- /.rhosts
- /etc/vfstab
- /etc/dfs/dfstab
- /etc/ftpd/ftpusers

ASET は、これらのファイルに関して各種の検査と変更を実行し、問題を sysconf.rpt ファイル内にレポートします。

環境変数の確認

このタスクでは、スーパーユーザー用とその他のユーザー用の PATH 環境変数と UMASK 環境変数がどのように設定されているかを検査します。この検査のために、/.profile、 /.login、 および /.cshrc ファイルがチェックされます。

環境のセキュリティー状況を検査した結果は、env.rpt ファイル内にレポートされます。

eeprom の確認

このタスクでは、eeprom セキュリティーパラメータの値が検査され、適切なセキュリティーレベルに設定されていることを確認します。eeprom セキュリティーパラメータは、 none、 command、 または full に設定できます。

ASET はこの設定を変更しませんが、推奨値を eeprom.rpt ファイル内にレポートします。

ファイアウォールの設定

このタスクでは、システムをネットワークリレーとして安全に使用できることが保証されます。62 ページの「ファイアウォールシステム」で説明しているように、このタスクでは、ファイアウォール専用システムが設定され、内部ネットワークが外部の公共ネットワークから保護されます。このファイアウォールシステムでは、ネットワークが 2 つに分割されます。このとき、分割された各ネットワークは、互いに信頼されないネットワークとして通信します。ファイアウォールの設定

タスクによって、インターネットプロトコル (IP) パケットを転送できなくなります。ファイアウォールによって、ルーティング情報も外部ネットワークから見えないように隠されます。

ファイアウォールのタスクはすべてのセキュリティーレベルで実行されますが、ファイアウォールとしての本来の機能は最上位レベルでのみ動作します。ASET を高セキュリティーレベルで実行するときでも、システムにはファイアウォール保護が不要であることがわかった場合は、asetenv ファイルを編集してファイアウォールタスクをはずすことができます。

行われた変更はすべて firewall.rpt ファイル内にレポートされます。

ASET 実行ログ

ASET を対話形式またはバックグラウンドで実行すると、実行ログが生成されます。デフォルトでは、ASET はログファイルを標準出力に生成します。実行ログは、ASET が指定された時刻に実行されたことを確認するもので、実行エラーメッセージも含まれています。aset -n コマンドを使用すると、ログを指定したユーザーに電子メールで配信できます。ASET オプションの一覧は、aset(1M) のマニュアルページを参照してください。

ASET 実行ログファイルの例

```
ASET running at security level low

Machine=example; Current time = 0325_08:00


aset: Using /usr/aset as working directory

Executing task list...
        firewall
        env
        sysconfig
        usrgrp
        tune
        cklist
        eeprom
All tasks executed. Some background tasks may still be running.

Run /usr/aset/util/taskstat to check their status:
     $/usr/aset/util/taskstat     aset_dir
Where aset_dir is ASET's operating directory, currently=/usr/aset

When the tasks complete, the reports can be found in:
     /usr/aset/reports/latest/*.rpt
You can view them by:
more /usr/aset/reports/latest/*.rpt
```

実行ログはまず、ASET が実行されたシステムと時刻を示します。次に、開始したタスクの一覧を表示します。

161 ページの「ASET のタスク一覧」で説明しているように、ASET はこれらのタスクごとにバックグラウンド処理を呼び出します。タスクは開始されると実行ログに一覧表示されます。この一覧は、タスクの完了を示しているわけではありません。バックグラウンドタスクの状態を確認するには、`taskstat` コマンドを使用します。

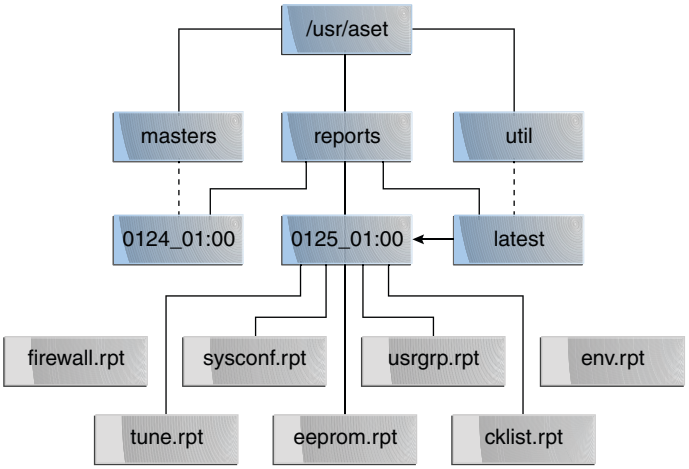
ASET レポート

ASET タスクから生成されたすべてのレポートファイルは、ディレクトリ `/usr/aset/reports` の下のサブディレクトリに入っています。この節では、`/usr/aset/reports` ディレクトリの構造と、レポートファイルを管理するためのガイドラインについて説明します。

ASET は、指定されたサブディレクトリにレポートファイルを格納し、レポートの生成日時を反映させます。この規則によって、ASET を実行するたびに変わるシステムの状態が記録されたレコードを順番に追跡できます。これらのレポートを監視し、比較して、システムセキュリティーの状況を判断できます。

次の図に `reports` ディレクトリ構造の例を示します。

図 7–1　ASET reports ディレクトリの構造



この例では、2 つのレポートサブディレクトリを示しています。

- `0124_01:00`
- `0125_01:00`

サブディレクトリ名は、レポートの生成日時を示します。各レポートサブディレクトリ名の形式は次のとおりです。

monthdate_hour:minute

この場合、*month*、 *date*、 *hour*、 *minute* は、いずれも 2 桁の数値です。たとえば、`0125_01:00` は 1 月 25 日の午前 1 時を表します。

2 つのレポートサブディレクトリには、それぞれ ASET を 1 度実行して、生成されたレポートの集合が含まれています。

`latest` ディレクトリは、常に最新レポートが入っているサブディレクトリを指すシンボリックリンクです。したがって、`/usr/aset/reports/latest` ディレクトリに移動すると、ASET で生成された最新レポートを見ることができます。このディレクトリには、最後に ASET を実行した各タスクのレポートファイルが入っています。

ASET レポートファイルの形式

各レポートファイルには、レポートを生成したタスクに基づいて名前が付けられます。次の表にタスクとそのレポートのリストを示します。

表 7–1 ASET のタスクと生成されるレポート

タスク	レポート
システムファイルのアクセス権の調整 (`tune`)	`tune.rpt`
システムファイルの確認 (`cklist`)	`cklist.rpt`
ユーザーとグループの確認 (`usrgrp`)	`usrgrp.rpt`
システム構成ファイルの確認 (`sysconf`)	`sysconf.rpt`
環境変数の確認 (`env`)	`env.rpt`
`eeprom` の確認 (`eeprom`)	`eeprom.rpt`
ファイアウォールの設定 (`firewall`)	`firewall.rpt`

各レポートファイル内で、メッセージの前後はバナー行で囲まれています。ASET の構成要素を誤って削除したり損傷したりすると、タスクが途中で終了することがあります。このような場合、通常はレポートファイルの末尾の方に、途中で終了した理由を示すメッセージが記録されています。

次にレポートファイル `usrgrp.rpt` の例を示します。

```
*** Begin User and Group Checking ***

Checking /etc/passwd ...
Warning! Password file, line 10, no passwd
:sync::1:1::/:/bin/sync
..end user check; starting group check ...
Checking /etc/group...
*** End User And group Checking ***
```

ASET レポートファイルの検査

ASET を最初に実行したときまたは再構成したときは、レポートファイルを詳細に検査する必要があります。再構成には、asetenv ファイル、masters サブディレクトリのマスターファイルを変更したり、ASET が動作するセキュリティーレベルを変更したりすることが含まれます。

このレポートには、ASET の再構成によって発生したエラーがすべて記録されます。レポートを詳しく確認すると、問題が発生した時点で対処して解決できます。

ASET レポートファイルの比較

構成上の変更やシステム更新がない期間中にレポートファイルを監視すると、レポートの内容が安定することがわかります。予期しなかった情報がレポートに含まれる場合は、diff ユーティリティーを使用してレポートを比較できます。

ASET マスターファイル

ASET のマスターファイル tune.high、 tune.low、 tune.med、 および uid_aliases は、ディレクトリ /usr/aset/masters に入っています。ASET は、マスターファイルを使用してセキュリティーレベルを定義します。詳細は、asetmasters(4) のマニュアルページを参照してください。

調整ファイル

tune.low、 tune.med、 tune.high マスターファイルでは、利用できる ASET セキュリティーレベルが定義されます。各ファイルでは、各レベルのシステムファイルの属性が指定され、比較と参照に使用されます。

uid_aliases ファイル

uid_aliases ファイルには、同じユーザー ID (UID) を共有する複数のユーザーアカウントの一覧が入っています。このような複数のユーザーアカウントがあると責任の所在があいまいになるため、通常は ASET が警告を出します。uid_aliases ファイル内で例外を一覧すると、この規則に例外を設けることができます。重複する UID を持つ passwd ファイル内のエントリを uid_aliases ファイル内で指定しておくと、これらのエントリは ASET でレポートされません。

複数のユーザーアカウントで同じ UID を共有しないでください。他の方法で目的を達成することを検討する必要があります。たとえば、複数のユーザーにアクセス権一式を共有させたい場合は、グループアカウントを作成できます。役割を作成することも可能です。UID の共有は、最後の手段としてほかの方法では目的を達成できない場合にだけ使用すべきです。

UID_ALIASES 環境変数を使用すると、別の別名ファイルを指定できます。デフォルトファイルは /usr/aset/masters/uid_aliases です。

確認リストファイル

システムファイルの確認に使用されるマスターファイルは、初めて ASET を実行するときか、セキュリティーレベルの変更後に ASET を実行するときに生成されます。

次の環境変数には、このタスクで確認するファイルを定義します。

- CKLISTPATH_LOW
- CKLISTPATH_MED
- CKLISTPATH_HIGH

ASET 環境ファイル (asetenv)

環境ファイル asetenv には、ASET タスクに影響する環境変数の一覧が入っています。一部の環境変数を変更すると、ASET の動作を修正することができます。asetenv ファイルの詳細は、asetenv(4) を参照してください。

ASET の構成

この節では、ASET を構成する方法と、ASET が稼働する環境について説明します。

ASET の管理と構成は最小限ですみ、ほとんどの場合はデフォルト値で実行できます。ただし、ASET の処理や動作に影響する一部のパラメータを調整して、その特長を最大限に発揮させることができます。デフォルト値を変更する前に、ASET の機能と、ASET がシステムの構成要素に及ぼす影響を理解しておく必要があります。

ASET は、次の 4 つの構成ファイルに依存してタスクの動作を制御します。

- /usr/aset/asetenv
- /usr/aset/masters/tune.low
- /usr/aset/masters/tune.med
- /usr/aset/masters/tune.high

環境ファイルの変更 (asetenv)

/usr/aset/asetenv ファイルには、2 つのメインセクションがあります。

- ユーザーが構成可能な環境変数セクション
- 内部環境変数セクション

ユーザーが構成可能なパラメータセクションは変更できます。ただし、内部環境変数セクションの設定は内部使用だけに限られ、変更できません。

ユーザーが構成可能なセクションのエントリを編集して、次の操作を行うことができます。

- 実行するタスクを選択する

- システムファイルの確認タスク用のディレクトリを指定する
- ASET の実行スケジュールを指定する
- UID 別名ファイルを指定する
- 確認対象を NIS+ テーブルまで拡張する

実行するタスクの選択: TASKS

ASET が実行する各タスクでは、システムセキュリティーの特定の領域が監視されます。ほとんどのシステム環境では、すべてのタスクでバランスがとれたセキュリティー範囲を提供する必要があります。ただし、1 つまたは複数のタスクを除外しても構いません。

たとえば、ファイアウォールタスクはすべてのセキュリティーレベルで実行されますが、本来の機能は最上位レベルでのみ動作します。ASET を高セキュリティーレベルで実行する場合でも、ファイアウォール保護は不要なときがあります。

管理者は、ファイアウォール機能を使用しないで高セキュリティーレベルで実行するように ASET を設定できます。このためには、`asetenv` ファイル内で環境変数の `TASKS` の一覧を編集します。デフォルトでは、`TASKS` の一覧にはすべての ASET タスクが含まれています。特定のタスクを削除するには、このファイルからそのタスクに関連する変数を削除します。この場合は、一覧から `firewall` 環境変数を削除することになります。次の一覧に ASET を実行すると、除外したタスクは実行されません。

次の例では、すべての ASET タスクが含まれる `TASKS` 一覧が表示されます。

```
TASKS=”env sysconfig usrgrp tune cklist eeprom firewall”
```

システムファイル確認タスクのディレクトリの指定: CKLISTPATH

システムファイル確認では、選択したシステムディレクトリ内のファイルの属性が確認されます。次の環境変数を使用して、どのディレクトリを確認するかを定義できます。

`CKLISTPATH_LOW` 変数は、低セキュリティーレベルで確認されるディレクトリを定義します。`CKLISTPATH_MED` と `CKLISTPATH_HIGH` 環境変数は、それぞれ中セキュリティーレベルと高セキュリティーレベルで同じように機能します。

セキュリティーレベルの低い環境変数を定義したディレクトリの一覧は、次にセキュリティーレベルの高い環境変数を定義したディレクトリの一覧のサブセットである必要があります。たとえば、`CKLISTPATH_LOW` に指定したディレクトリはすべて `CKLISTPATH_MED` に含めます。同様に、`CKLISTPATH_MED` に指定したディレクトリはすべて `CKLISTPATH_HIGH` に含めます。

これらのディレクトリに対して実行される確認は再帰的ではありません。ASET では、環境変数に明示的に指定されているディレクトリだけが確認されます。ASET では、そのサブディレクトリは確認されません。

これらの環境変数の定義を編集して、ASET に確認させたいディレクトリを追加または削除できます。これらの確認リストは、通常は毎日変更がないシステムファイルにのみ便利なことに注意してください。たとえば、一般にユーザーのホームディレクトリは動的な変化が大きすぎるので、確認リストの候補にはなりません。

ASET の実行スケジュールの指定: PERIODIC_SCHEDULE

ASET は対話形式で実行できます。また、-p オプションを使用して、ASET タスクの実行を指定した時刻に要求することもできます。ASET は、システム需要が少ないときに定期的に実行できます。たとえば、ASET は PERIODIC_SCHEDULE を照会して、ASET タスクの実行頻度と実行時刻を判断します。ASET を定期的に実行するように設定する方法については、178 ページの「ASET を定期的に実行する方法」を参照してください。

PERIODIC_SCHEDULE の形式は、crontab エントリの形式と同じです。詳細は、crontab(1) を参照してください。

別名ファイルの指定: UID_ALIASES

UID_ALIASES 変数は、共有 UID がリストされる別名ファイルを指定します。デフォルトファイルは /usr/aset/masters/uid_aliases です。

確認範囲を NIS+ テーブルまで拡張: YPCHECK

YPCHECK 環境変数は、ASET でシステム構成ファイルテーブルも確認するかどうかを指定します。YPCHECK はブール型変数です。YPCHECK には true または false しか指定できません。デフォルト値は false で、NIS+ テーブルの確認は無効になっています。

この環境変数の機能を理解するために、passwd ファイルに与える影響を考えてみてください。false に設定すると、ASET はローカルの passwd ファイルを確認します。true に設定すると、NIS+ の passwd テーブル内でシステムのドメインも確認されます。

注 – ASET ではローカルファイルが自動的に修復されますが、NIS+ テーブル内の潜在的な問題はレポートされるだけです。NIS+ テーブルの変更は行いません。

調整ファイルの変更

ASET は、3 つのマスター調整ファイル tune.low、 tune.med、 tune.high を使用して、重要なシステムファイルへのアクセス制限を緩めたり厳しくしたりします。この 3 つのマスターファイルは /usr/aset/masters ディレクトリにあり、環境に合わせて調整できます。詳細は、175 ページの「調整ファイルの例」を参照してください。

tune.low ファイルは、アクセス権をデフォルトのシステム設定に適した値に設定します。tune.med ファイルは、これらのアクセス権をさらに制限し、tune.low に含まれていないエントリを追加します。tune.high ファイルは、アクセス権をさらに厳しく制限します。

注 – 調整ファイル内の設定を変更するには、ファイルのエントリを追加または削除します。アクセス権を現在の設定よりも制限が緩やかになるような値に設定しても意味がありません。システムセキュリティーを下位レベルに下げない限り、ASET がアクセス権の制限を緩和したことになりません。

ASET で変更されたシステムファイルの復元

ASET を初めて実行すると、元のシステムファイルが保存され保管されます。`aset.restore` ユーティリティーは、これらのファイルを復元します。また、ASET を定期的に実行するようにスケジュール指定している場合は、そのスケジュールを解除します。`aset.restore` コマンドは、ASET の操作ディレクトリ `/usr/aset` に入っています。

システムファイルに適用した変更は、`aset.restore` コマンドを実行すると失われます。

`aset.restore` コマンドは、次のような場合に使用します。

- ASET の変更を削除して元のシステムを復元する場合。

  ASET を永久に無効にする場合は、以前にスーパーユーザーの `crontab` に `aset` コマンドが追加されていれば、`cron` スケジュールから ASET を削除できます。`cron` を使用して自動実行を削除する方法については、179 ページの「ASET の定期的な実行を停止する方法」を参照してください。
- ASET を短期間実行したあとに、元のシステム状態を復元する場合。
- 一部の主要なシステム機能が正常に動作せず、ASET が原因だと思われる場合。

NFS システムを使用するネットワーク操作

通常、ネットワークの一部となっているシステム上でも、ASET はスタンドアロンモードで使用されます。スタンドアロンシステムのシステム管理者は、システムのセキュリティーを守る必要があるため、ASET の稼働と管理を通してシステム保護に当たります。

また、ASET は NFS 分散環境でも使用できます。ネットワーク管理者は、すべてのクライアントの各種管理タスクのインストール、実行、管理を担当します。複数のクライアントシステムにわたって ASET を管理しやすくするため、すべてのクライアントに一括して適用されるような構成変更を行えます。変更を一括して適用すると、各システムにログインして構成変更を繰り返す必要がなくなります。

ネットワークシステム上で ASET の設定方法を決めるときには、セキュリティー制御を誰に行わせるかを考える必要があります。たとえば、ユーザーに各自のシステムにおけるセキュリティーの一部を制御させることができます。また、セキュリティー制御の責任を 1 つにまとめることもできます。

各セキュリティーレベルの一括構成の提供

複数のネットワーク構成を設定する場合があります。たとえば、低セキュリティーレベルに設定したクライアント用に 1 つ、中レベルのクライアント用に 1 つ、さらに高レベルのクライアント用に 1 つのネットワーク構成を設定できます。

セキュリティーレベルごとに個別の ASET ネットワーク構成を作成する場合は、サーバー上に 3 つの ASET 構成を作成できます。この場合、レベルごとに 1 つずつ構成を作成することになります。各構成を該当するセキュリティーレベルのクライアントにエクスポートします。3 つの構成すべてに共通の ASET 構成要素は、リンクを使用して共有できます。

ASET レポートの収集

管理者は、サーバー上に ASET 構成要素を集中できるだけでなく、サーバー上に集中ディレクトリを設定してすべてのレポートを収集できます。クライアントは、スーパーユーザー特権のあるなしにかかわらずサーバーにアクセスできます。収集メカニズムを設定する方法については、180 ページの「サーバー上で ASET レポートを収集する方法」を参照してください。

サーバー上でレポートを収集するように設定すると、すべてのクライアントに関するレポートを 1 箇所で検討できます。この方法は、クライアントがスーパーユーザー特権を持っているかどうかに関係なく使用できます。また、ユーザーに各自の ASET レポートを監視させたい場合は、ローカルシステム上にレポートディレクトリを残しておくこともできます。

ASET 環境変数

次の表にASET 環境変数と各変数で指定する値を示します。

`ASETDIR`	ASET の作業ディレクトリを指定します
`ASETSECLEVEL`	セキュリティーレベルを指定します
`PERIODIC_SCHEDULE`	定期的なスケジュールを指定します
`TASKS`	実行する ASET タスクを指定します
`UID_ALIASES`	別名ファイルを指定します
`YPCHECK`	NIS マップと NIS+ テーブルを確認するかどうかを指定します
`CKLISTPATH_LOW`	低セキュリティー用のディレクトリリストです
`CKLISTPATH_MED`	中セキュリティー用のディレクトリリストです
`CKLISTPATH_HIGH`	高セキュリティー用のディレクトリリストです

次の節で示す環境変数は、/usr/aset/asetenv ファイルにあります。ASETDIR と ASETSECLEVEL 変数はオプションです。これらの変数は、シェルから /usr/aset/aset コマンドを使用しなければ設定できません。他の環境変数は、ファイルを編集して設定できます。

ASETDIR 環境変数

ASETDIR は、ASET の作業ディレクトリを指定します。

C シェルからは、次のように入力します。

```
% setenv ASETDIR pathname
```

Bourne シェルまたは Korn シェルからは、次のように入力します。

```
$ ASETDIR=pathname
$ export ASETDIR
```

pathname には ASET 作業ディレクトリの完全パス名を設定します。

ASETSECLEVEL 環境変数

ASETSECLEVEL 変数は、ASET タスクが実行されるセキュリティーレベルを指定します。

C シェルからは、次のように入力します。

```
% setenv ASETSECLEVEL level
```

Bourne シェルまたは Korn シェルからは、次のように入力します。

```
$ ASETSECLEVEL=level
$ export ASETSECLEVEL
```

上記のコマンドで、*level* を次のいずれかに設定できます。

`low` 低セキュリティーレベル

`med` 中セキュリティーレベル

`high` 高セキュリティーレベル

PERIODIC_SCHEDULE 環境変数

PERIODIC_SCHEDULE の値の形式は、crontab ファイルと同じです。変数の値は二重引用符で囲んだ 5 つのフィールドからなる文字列として指定します。各フィールドは次のように 1 つの空白文字で区切ります。

"*minutes hours day-of-month month day-of-week*"

minutes hours 開始時刻を (0 から 59) と時間 (0 から 23) で指定します。

day-of-month ASET を実行する日付を 1 から 31 の値で指定します。

month ASET を実行する月を 1 から 12 の値で指定します。

day-of-week ASET を実行する曜日を 0 から 6 の値で指定します。日曜日が 0 になります。

ASET の定期的なスケジュールを作成するときは、次の規則が適用されます。

- どのフィールドも、値のリストをコンマで区切って指定できます。
- 値を数値または範囲として指定できます。範囲は、一対の数値をハイフンで結合して指定します。範囲に含まれるすべての時刻に ASET タスクを実行することを示します。
- どのフィールドも、値としてアスタリスク(*) を指定できます。アスタリスクを指定すると、そのフィールドで使用できるすべての値を指定したことになります。

PERIODIC_SCHEDULE 変数のデフォルトエントリでは、ASET が毎日深夜 12:00 に実行されます。

```
PERIODIC_SCHEDULE="0 0 * * *"
```

TASKS 環境変数

TASKS 変数は、ASET で実行されるタスクを一覧表示します。デフォルトでは、7 つのタスクが次のようにすべて一覧されます。

```
TASKS="env sysconfig usrgrp tune cklist eeprom firewall"
```

UID_ALIASES 環境変数

UID_ALIASES 変数は、別名ファイルを指定します。別名ファイルがあると、ASET は使用可能な複数の別名の一覧をこのファイル内で照会します。書式は UID_ALIASES=*pathname* です。*pathname* は、別名ファイルのフルパス名です。

デフォルトは次のとおりです。

```
UID_ALIASES=${ASETDIR}/masters/uid_aliases
```

YPCHECK 環境変数

YPCHECK 変数は、システムテーブルを確認するタスクを拡張して NIS または NIS+ テーブルを含めます。YPCHECK 変数はブール変数なので、true または false に設定されます。

デフォルトは false で、ローカルシステムテーブルが確認されます。

```
YPCHECK=false
```

CKLISTPATH_level 環境変数

3 つの確認リストパス変数は、システムファイルの確認タスクで確認されるディレクトリを一覧表示します。次の変数の定義は、デフォルトで設定されます。これらの定義は、さまざまなレベルの変数の関係を示しています。

```
CKLISTPATH_LOW=${ASETDIR}/tasks:${ASETDIR}/util:${ASETDIR}/masters:/etc
CKLISTPATH_MED=${CKLISTPATH_LOW}:/usr/bin:/usr/ucb
CKLISTPATH_HIGH=${CKLISTPATH_MED}:/usr/lib:/sbin:/usr/sbin:/usr/ucblib
```

確認リストパス環境変数の値は、シェルパス変数の値と似ています。シェルパス変数と同様に、確認リストパス環境変数はディレクトリ名のリストです。ディレクトリ名はコロンで区切ります。等号 (=) を使用すると、変数名にその値を設定できます。

ASET ファイルの例

この節では、調整ファイルや別名ファイルなどの ASET ファイルの例を示します。

調整ファイルの例

ASET は 3 つの調整ファイルを管理します。調整ファイル内の各エントリは 1 行に入っています。エントリ内のフィールドの順序は次のとおりです。

pathname mode owner group type

pathname	ファイルのフルパス名
mode	アクセス権の設定を表す 5 桁の数値
owner	ファイルの所有者
group	ファイルのグループ
type	ファイルの形式

調整ファイルを編集するときは、次の規則が適用されます。

- アスタリスク (*) や疑問符 (?) などの標準シェルワイルドカード文字をパス名に使用することにより、複数のファイルを参照できます。詳細は、sh(1) を参照してください。
- *mode* は、もっとも制限が緩やかな値を表します。現在の設定が指定した値よりもすでに厳しく制限されている場合、ASET はアクセス権の設定を緩和しません。たとえば、指定した値が **00777** の場合、**00777** は常に現在の設定よりも緩やかな制限を表すため、アクセス権は変更されません。

  このプロセスは、ASET がモード設定をどのように処理するかというものです。セキュリティーレベルのレベルが低いものに変更されるか、あるいは管理者が ASET を削除する場合は、別のプロセスとなります。セキュリティーレベルを前回の実行時よりも下げるときや、ASET が最初に実行される前の状態のシステムファイルを復元するときには、ASET は操作の内容を認識して保護レベルを下げます。
- *owner* と *group* には、数値 ID ではなく名前を使用する必要があります。
- *owner*、*group*、*type* の代わりに疑問符 (?) を使用すると、ASET がこれらのパラメータの既存の値を変更しないようにします。
- *type* には、`symlink` (シンボリックリンク)、ディレクトリ、またはファイルを指定できます。
- セキュリティーレベルが高くなるほど、調整ファイルは下位レベルよりも緩やかなファイルアクセス権にリセットされます。また、上位セキュリティーレベルになるほど、一覧に多数のファイルが追加されます。
- 1 つのファイルで複数の調整ファイルエントリを照合できます。たとえば、`etc/passwd` は `etc/pass*` エントリと `/etc/*` エントリに一致します。
- 2 つのエントリのアクセス権が異なる場合は、ファイルアクセス権はもっとも厳しいアクセス権を表す値に設定されます。次の例では、`/etc/passwd` ファイルのアクセス権は **00755** に設定されますが、これは **00755** は **00770** よりも厳密な制限であることを表します。

  ```
  /etc/pass* 00755 ? ? file
  /etc/* 00770 ? ? file
  ```
- 2 つのエントリの *owner* 指定または *group* 指定が異なる場合は、最後のエントリが優先されます。次の例では、`/usr/sbin/chroot` の所有者が `root` に設定されます。

  ```
  /usr/sbin/chroot 00555 bin bin file
  /usr/sbin/chroot 00555 root bin file
  ```

別名ファイルの例

別名ファイルには、同じユーザー ID を共有する別名の一覧が含まれています。

各エントリの書式は次のとおりです。

uid=alias1 =alias2=alias3 =. . .

uid 共有 UID。

aliasn UID を共有するユーザーアカウント。

たとえば、次のエントリでは UID `0` を一覧表示しています。この UID は、`sysadm` および `root` アカウントによって共有されています。

```
0=root=sysadm
```

ASET の実行 (作業マップ)

作業	説明	参照先
ASET をコマンド行から実行します	指定する ASET レベルにあるシステムを保護します。実行ログを表示して変化を確認します	177 ページの「ASET を対話的に実行する方法」
定期的に ASET をバッチモードで実行します	cron ジョブを設定し、ASET がシステムを保護するようにします。	178 ページの「ASET を定期的に実行する方法」
バッチモードによる ASET の実行を停止します	ASET cron ジョブを削除します。	179 ページの「ASET の定期的な実行を停止する方法」
ASET レポートをサーバーに保存します	監視のためにクライアントから ASET レポートを収集し、中心的な場所に保存します。	180 ページの「サーバー上で ASET レポートを収集する方法」

ASET で変数を設定する方法は、172 ページの「ASET 環境変数」を参照してください。ASET を構成する方法は、168 ページの「ASET の構成」を参照してください。

▼ ASET を対話的に実行する方法

1 スーパーユーザーになるか、同等の役割を引き受けます。

役割には、認証と特権コマンドが含まれます。役割の詳細は、210 ページの「RBAC の構成 (作業マップ)」を参照してください。

2 **aset** コマンドを使用して **ASET** を対話的に実行します。

```
# /usr/aset/aset -l level -d pathname
```

level セキュリティーレベルを指定します。有効な値は `low`、`medium`、または `high` です。デフォルト設定は `low` です。セキュリティーレベルの詳細は、160 ページの「ASET のセキュリティーレベル」を参照してください。

pathname ASET の作業ディレクトリを指定します。デフォルトは /usr/aset です。

3 画面に表示される **ASET** 実行ログを見て、**ASET** が動作していることを確認します。
実行ログメッセージは、動作しているタスクを示します。

例 **7–1** ASET を対話的に実行する

次の例では、デフォルトの作業ディレクトリを使用して低セキュリティーレベルで ASET を実行します。

```
# /usr/aset/aset -l low
======= ASET Execution Log =======

ASET running at security level low

Machine = jupiter; Current time = 0111_09:26

aset: Using /usr/aset as working directory

Executing task list ...
    firewall
    env
    sysconf
    usrgrp
    tune
    cklist
    eeprom

All tasks executed. Some background tasks may still be running.

Run /usr/aset/util/taskstat to check their status:
 /usr/aset/util/taskstat [aset_dir]

where aset_dir is ASET's operating
directory,currently=/usr/aset.

When the tasks complete, the reports can be found in:
 /usr/aset/reports/latest/*.rpt

You can view them by:
 more /usr/aset/reports/latest/*.rpt
```

▼ ASET を定期的に実行する方法

1 スーパーユーザーになるか、同等の役割を引き受けます。
役割には、認証と特権コマンドが含まれます。役割の詳細は、210 ページの「RBAC の構成 (作業マップ)」を参照してください。

2 必要であれば、**ASET** を定期的に実行する時刻を設定します。

システム需要が少ないときに ASET を実行します。/usr/aset/asetenv ファイル内の PERIODIC_SCHEDULE 環境変数を使用して、ASET を定期的に実行する時刻を設定します。デフォルトでは、時刻は深夜に設定されます。

別の時刻を設定する場合は、/usr/aset/asetenv ファイル内の PERIODIC_SCHEDULE 変数を編集します。PERIODIC_SCHEDULE 変数の設定の詳細は、173 ページの「PERIODIC_SCHEDULE 環境変数」を参照してください。

3 **aset** コマンドを使用してエントリを **crontab** ファイルに追加します。

```
# /usr/aset/aset -p
```

-p オプションは、決めた時刻に ASET の実行を開始するように /usr/aset/asetenv ファイル内の PERIODIC_SCHEDULE 環境変数に設定した行を crontab ファイルに挿入します。

4 次のコマンドを実行すると **crontab** エントリが表示され、**ASET** の実行スケジュールを確認できます。

```
# crontab -l root
```

▼ ASET の定期的な実行を停止する方法

1 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 **crontab** ファイルを編集します。

```
# crontab -e root
```

3 **ASET** エントリを削除します。

4 変更を保存して終了します。

5 **crontab** エントリを表示して、**ASET** エントリが削除されていることを確認します。

```
# crontab -l root
```

▼ サーバー上で ASET レポートを収集する方法

1 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 サーバー上でディレクトリを設定します。

a. **/usr/aset** ディレクトリに移動します。

```
mars# cd /usr/aset
```

b. *rptdir* ディレクトリを作成します。

```
mars# mkdir rptdir
```

c. *rptdir* ディレクトリに移動して、*client_rpt* ディレクトリを作成します。

これにより、クライアント用のサブディレクトリ *client_rpt* が作成されます。レポートを収集するクライアントごとに、この手順を繰り返します。

```
mars# cd rptdir
mars# mkdir client_rpt
```

次の例では、ディレクトリ `all_reports` とサブディレクトリ `pluto_rpt` および `neptune_rpt` が作成されます。

```
mars# cd /usr/aset
mars# mkdir all_reports
mars# cd all_reports
mars# mkdir pluto_rpt
mars# mkdir neptune_rpt
```

3 *client_rpt* ディレクトリを **/etc/dfs/dfstab** ファイルに追加します。

このディレクトリには、読み取り権と書き込み権があります。

たとえば、`dfstab` ファイル内の次のエントリは、読み取り権と書き込み権によって共有されます。

```
share -F nfs -o rw=pluto /usr/aset/all_reports/pluto_rpt
share -F nfs -o rw=neptune /usr/aset/all_reports/neptune_rpt
```

4 **dfstab** ファイル内のリソースをクライアントが利用できるようにします。

```
# shareall
```

5 各クライアント上でクライアントのサブディレクトリを、マウントポイント **/usr/aset/masters/reports** にサーバーからマウントします。

```
# mount server:/usr/aset/client_rpt /usr/aset/masters/reports
```

6 **/etc/vfstab**ファイルを編集して、ブート時にディレクトリを自動的にマウントするようにします。

neptune 上の /etc/vfstab 内の次のエントリ例には、mars からマウントされるディレクトリ /usr/aset/all_reports/neptune_rpt と、neptune 上のマウントポイント /usr/aset/reports が一覧されています。ブート時には、vfstab 内に一覧されたディレクトリが自動的にマウントされます。

```
mars:/usr/aset/all_reports/neptune.rpt /usr/aset/reports nfs - yes hard
```

ASETの問題の障害追跡

この節では、ASET によって生成されるエラーメッセージについて説明します。

ASETのエラーメッセージ

`ASET failed: no mail program found.`
原因: ASET は実行ログをユーザーに送るように指示されましたが、メールプログラムが見つかりません。

対処方法: メールプログラムをインストールしてください。

`Usage: aset [-n user[@host]] in /bin/mail or /usr/ucb/mail.`

`Cannot decide current and previous security levels.`
原因: ASET は、今回と前回の呼び出しのセキュリティーレベルを判別できません。

対処方法: 現在のセキュリティーレベルがコマンド行オプションまたは ASETSECLEVEL 環境変数によって設定されているかどうかを確認してください。また、ASETDIR/archives/asetseclevel.arch の最終行に、以前のセキュリティーレベルが正しく反映されているかどうかを確認してください。これらの値が設定されていないか正しくない場合は、正しい値を入力してください。

`ASET working directory undefined.`

`To specify, set ASETDIR environment variable or use command line option -d.`

`ASET startup unsuccessful.`
原因: ASET の作業ディレクトリが定義されていないか、正しく定義されていません。

対処方法: ASETDIR 環境変数または -d コマンド行オプションを使用してエラーを訂正してから、ASET を再起動してください。

`ASET working directory $ASETDIR missing.`

`ASET startup unsuccessful.`

原因: ASETの作業ディレクトリが定義されていないか、正しく定義されていません。`ASETDIR`変数または`-d`コマンド行オプションによって、存在しないディレクトリが参照されている可能性があります。

対処方法: 正しいディレクトリ、つまりASETディレクトリ階層が入っているディレクトリが正しく参照されているかどうかを確認してください。

`Cannot expand $ASETDIR to full pathname.`

原因: ASETが`ASETDIR`変数または`-d`コマンド行オプションで指定されたディレクトリ名を完全パス名に展開できません。

対処方法: ディレクトリ名が正しいか確認します。ユーザーがアクセス権を持つ既存のディレクトリをそのディレクトリが参照しているか確認してください。

`aset: invalid/undefined security level.`

`To specify, set ASETSECLEVEL environment variable or use command line option -l, with argument= low/med/high.`

原因: セキュリティーレベルが定義されていないか、正しく定義されていません。`low`、 `med`、 または`high`の値以外は定義できません。

対処方法: `ASETSECLEVEL`変数または`-l`コマンド行オプションを使用して、low、med、 またはhighのいずれかの値を指定してください。

`ASET environment file asetenv not found in $ASETDIR.`

`ASET startup unsuccessful.`

原因: ASETは`asetenv`ファイルを作業ディレクトリ内で見つけることができません。

対処方法: ASETの作業ディレクトリ内に`asetenv`ファイルが入っているかどうかを確認してください。このファイルの詳細は、`asetenv(4)`のマニュアルページを参照してください。

`filename doesn't exist or is not readable.`

原因: *filename*で指定されたファイルが存在しないか、読み取れません。この問題は、`-u`オプションを使用している場合に発生します。このオプションは確認するユーザーの一覧が入ったファイルを指定するために使用します。

対処方法: `-u`オプションの引数が存在することと、その引数が読み取り可能であるかを確認してください。

`ASET task list TASKLIST undefined.`
原因: asetenv ファイル内で定義されているはずの ASET タスクリストが定義されていません。asetenv ファイルが無効である可能性があります。

対処方法: asetenv ファイルを検査してください。タスクリストが User Configurable セクションで定義されているかどうかを確認します。また、ファイルの他の部分をチェックして、ファイルが変更されていないことを確認します。有効な asetenv ファイルの内容については、asetenv(4) のマニュアルページを参照してください。

`ASET task list $TASKLIST missing.`

`ASET startup unsuccessful.`
原因: asetenv ファイル内で定義されているはずの ASET タスクリストが定義されていません。asetenv ファイルが無効である可能性があります。

対処方法: asetenv ファイルを検査してください。タスクリストが User Configurable セクションで定義されているかどうかを確認します。また、ファイルの他の部分をチェックして、ファイルが変更されていないことを確認します。有効な asetenv ファイルの内容については、asetenv(4) のマニュアルページを参照してください。

`Schedule undefined for periodic invocation.`

`No tasks executed or scheduled. Check asetenv file.`
原因: -p オプションを使用して ASET のスケジュール指定が要求されましたが、環境変数 PERIODIC_SCHEDULE が asetenv ファイル内で定義されていません。

対処方法: asetenv ファイルの User Configurable セクションをチェックして、変数が定義されていることを確認してください。また、変数の書式が正しいかも確認してください。

`Warning! Duplicate ASET execution scheduled.`

`Check crontab file.`
原因: ASET のスケジュールが複数回指定されています。つまり、ASET スケジュールがまだ有効な間に別のスケジュールを指定するように要求されています。複数のスケジュールが必要な場合は、このメッセージはエラーを示すものではなく、警告メッセージとなります。複数のスケジュールが必要な場合は、crontab コマンドを使用して、正しいスケジュール書式を使用する必要があります。詳細は、crontab(1) のマニュアルページを参照してください。

対処方法: crontab コマンドを使用して、正しいスケジュールが有効になっていることを検証してください。ASET に関して不要な crontab エントリがないかどうかを確認してください。

パート III

役割、権利プロファイル、特権

このパートでは、役割によるアクセス制御 (RBAC) とプロセス権管理について説明します。RBAC コンポーネントには、役割、権利プロファイル、承認があります。プロセス権管理は、特権を介して実装されます。特権を RBAC と連携すると、スーパーユーザーによるシステムの管理よりも安全に管理することができます。

◆ ◆ ◆ 第 8 章

8 役割と特権の使用 (概要)

Oracle Solaris の「役割によるアクセス制御 (RBAC)」と特権は、スーパーユーザーよりも厳密な機密保護機能です。この章では、RBAC と特権についての概要を述べます。

この章の内容は次のとおりです。

RBAC の新機能

Solaris 10 8/07: このリリースから、`project.max-locked-memory` および `zone.max-locked-memory` 資源制御が導入されました。`PRIV_PROC_LOCK_MEMORY` 特権をあるユーザーや非大域ゾーンに割り当てる場合に、これらの資源制御を使えば、そのユーザーやゾーンがすべてのメモリーをロックしてしまうのを防ぐことができます。詳細については、203 ページの「特権とシステム資源」を参照してください。

Solaris 10 10/08: このリリースでは、`solaris.admin.usermgr` 承認が再編成され、高度にセキュリティー保護されたインストールのセキュリティー要件である「責務の分離」をサポートするようになりました。責務の分離を満たすには、ユーザーアカウントの作成に 2 つのアカウントが必要になります。この要件に対応するようにソフトウェアを構成するには、『Oracle Solaris Trusted Extensions 構成ガイド』の「責務分離を実施する権利プロファイルを作成する」を参照してください。また、このリリースでは、役割のパスワードを変更する方法もこのドキュメントの 230 ページの「役割のパスワードを変更する方法」で説明されています。

Solaris 10 9/10: このリリースでは、特権の基本セットに `net_access` 特権が追加されました。特権については、`privileges(5)` のマニュアルページを参照してください。

役割によるアクセス制御 (概要)

役割によるアクセス制御 (RBAC) は、通常はスーパーユーザーに限定されているタスクに対するユーザーアクセスを制御するセキュリティー機能です。RBAC では、プロセスやユーザーにセキュリティー属性を適用することで、スーパーユーザーの権限を複数の管理者に分けることができます。プロセス権管理は、「特権」を介して実装されます。ユーザー権管理は RBAC によって行われます。

- プロセス権管理については、199 ページの「特権 (概要)」を参照してください。
- RBAC 関連の作業については、第 9 章「役割によるアクセス制御の使用 (手順)」を参照してください。
- 参照情報については、第 10 章「役割によるアクセス制御 (参照)」を参照してください。

RBAC: スーパーユーザーモデルの代替機能

従来の UNIX システムでは、`root` ユーザー (スーパーユーザーとも呼ばれる) が全権を有します。`root` として実行されるプログラム、または `setuid` プログラムにも全権があります。`root` ユーザーは全ファイルの読み取り権とアクセス権、全プログラムの実行権を持ち、任意のプロセスに終了シグナルを送ることができます。実際、スーパーユーザーになるユーザーは、使用するサイトのファイアウォールの変更、監査トレールの変更、機密レコードの読み取り、ネットワーク全体の停止などを行えます。`setuid` プログラムがハイジャック (強奪) されると、システム上で何が起きても不思議はありません。

役割によるアクセス制御 (RBAC) は、絶対的なスーパーユーザーモデルに代わるより厳密な機密保護機能です。RBAC を使用することで、セキュリティーポリシーをきめ細かく適用できます。RBAC では、「最小特権」というセキュリティー原則が使用されます。最小特権は、ジョブを行う上で必要な特権だけをユーザーに与えることを意味します。通常のユーザーには、アプリケーションの実行、実行中のジョブの状態チェック、ファイルの出力、新しいファイルの作成などに必要なだけの特権が与えられます。通常のユーザー権限を超える権限は、権利プロファイルとしてグループ化されます。スーパーユーザー権限の一部が必要なジョブを行うユーザーは、適切な権利プロファイルを含む役割を引き受けます。

RBAC では、スーパーユーザー権限を「権利プロファイル」としてまとめます。これらの権利プロファイルは、「役割」と呼ばれる特殊なユーザーアカウントに割り当てられます。ユーザーは、スーパーユーザー権限の一部を必要とするジョブを行う場合に役割を引き受けることができます。Oracle Solaris ソフトウェアには、あらかじめ定義された権利プロファイルが用意されています。管理者は役割を作成し、プロファイルを割り当てます。

権利プロファイルを使用することで、さまざまな権限を提供できます。たとえば、Primary Administrator 権利プロファイルはスーパーユーザーと同等です。権限を限定して権利プロファイルを定義することもできます。たとえば、Cron Management

権利プロファイルは at ジョブと cron ジョブを管理します。役割を作成する場合は、多くの権限を持つ役割を作ることも、わずかな権限を持つ役割を作ることもできます。あるいはこの両方を作成することも可能です。

RBAC モデルでは、スーパーユーザーが 1 つ以上の役割を作成します。役割は、権利プロファイルに基づいて作成されます。続いてスーパーユーザーは、役割の作業に適したユーザーにその役割を割り当てます。ユーザーは、各自のユーザー名でログインします。ログイン後ユーザーは、限定された管理コマンドとグラフィカルユーザーインタフェース (GUI) ツールを実行できる役割を引き受けます。

役割は柔軟に設定できるため、さまざまなセキュリティーポリシーに対応できます。Oracle Solaris には標準装備された役割がほとんどありませんが、推奨される 3 つの役割を簡単に構成できます。これらの役割は、同じ名前の権利プロファイルに基づいています。次にこれらの役割を示します。

- **Primary Administrator** – root ユーザー (スーパーユーザー) と同等の強力な役割。
- **root** – root ユーザーと同等の強力な役割。ただし、この root はログインを行うことはできません。通常のユーザーは、ログインしてから、割り当てられた root 役割を引き受ける必要があります。
- **System Administrator** – セキュリティーに関係のない管理作業を行う役割で、権限が限定されています。この役割ではファイルシステム、メール、ソフトウェアのインストールなどを管理できます。ただし、パスワードの設定は行えません。
- **Operator** – バックアップやプリンタ管理などが行える、補佐的な管理者向けの役割。

注 – Media Backup 権利プロファイルは、ルートファイルシステム全体へのアクセスを提供します。したがって、Media Backup 権利プロファイルと Operator 権利プロファイルは補佐的な管理者向けに設計されていますが、この管理者が信頼できるユーザーであることを確認する必要があります。

これらの 3 つの役割の実装は必須ではありません。役割は、組織のセキュリティー要件に応じて設定する機能です。役割は、セキュリティー、ネットワーク、ファイアウォールの管理など、特定の目的の管理に合わせて設定できます。別の方法として、強力な管理者役割を 1 つと上級ユーザー役割を作成することもできます。この上級ユーザー役割は、自分のシステムの各部を修正することを認められたユーザーに割り当てます。

スーパーユーザーモデルと RBAC モデルは共存できます。次の表は、RBAC モデルで設定できる権限 (スーパーユーザーから制限された通常ユーザーまで) を順に挙げ、両モデルで監視できる管理アクションを示しています。システム上での特権の効果だけをまとめた情報としては、表 8–2 を参照してください。

表 8–1 スーパーユーザーモデルと特権を使用した RBAC モデルの比較

システムにおけるユーザー権限	スーパーユーザーモデル	RBAC モデル
すべてのスーパーユーザー権限を持つスーパーユーザーになる	はい	はい
すべてのユーザー権限を持つユーザーとしてログインする	はい	はい
権限が限定されたスーパーユーザーになる	いいえ	はい
ユーザーとしてログインし、散発的にスーパーユーザー権限を持つ	可能 (setuid プログラムのみ)	可能 (setuid プログラム、および RBAC を使用)
すべてのスーパーユーザー権限ではなく、一部の管理権限だけを持つユーザーとしてログインする	いいえ	可能 (RBAC を使用、および直接割り当てられた特権と承認を使用)
通常のユーザーよりも少ない権限を持つユーザーとしてログインする	いいえ	可能 (RBAC を使用、および特権を削除してログイン)
スーパーユーザーの処理を監視する	可能 (su コマンドを監査することによって監視)	可能 (プロファイルシェルコマンドを監査することによって監視) root ユーザーが無効になっている場合には、root 役割を引き受けたユーザーの名前が監査トレールに入ります

Oracle Solaris RBAC の要素と基本概念

Oracle Solaris の RBAC モデルでは、次の要素が導入されました。

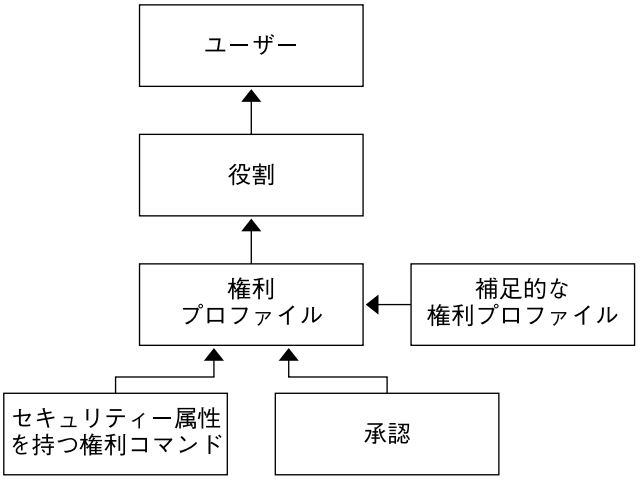
- 承認 – 追加権限を必要とするアクション群をユーザーまたは役割が実行できるようにするアクセス権。たとえばインストール時には、セキュリティーポリシーによって solaris.device.cdrw 承認が通常のユーザーに与えられます。この承認によってユーザーは CD-ROM デバイスの読み取りと書き込みが行えます。承認の一覧は、/etc/security/auth_attr ファイルを参照してください。
- 特権 – コマンド、ユーザー、役割、またはシステムに与えることができる個別の権利。特権によってプロセスの正常実行が可能となります。たとえば、proc_exec 特権によってプロセスは execve() を呼び出すことができます。通常のユーザーには基本特権が与えられます。自分の基本特権を確認するには、ppriv -vl basic コマンドを実行します。
- セキュリティー属性 – プロセス処理を可能にする属性。典型的な UNIX 環境では、通常のユーザーに禁止されているプロセス処理が、セキュリティー属性によって可能になります。たとえば、setuid プログラムと setgid プログラムはセキュリティー属性を持ちます。RBAC モデルでは、通常のユーザーが行う処理で

セキュリティー属性が要求されることがあります。RBAC モデルでは、setuid プログラムや setgid プログラムだけでなく、承認と特権もセキュリティー属性です。たとえば、solaris.device.allocate 承認が与えられたユーザーは、デバイスを独占的に使用するためにそのデバイスの割り当てを行うことができます。sys_time 特権を持つプロセスは、システム時刻を操作できます。

- 特権付きアプリケーション – セキュリティー属性を確認することによってシステム制御に優先するアプリケーションまたはコマンド。典型的な UNIX 環境と RBAC モデルでは、setuid と setgid を使用するプログラムは特権付きアプリケーションです。RBAC モデルでは、処理を正常に実行する上で特権または承認を必要とするプログラムも特権付きアプリケーションです。詳細は、195 ページの「特権付きアプリケーションと RBAC」を参照してください。
- 権利プロファイル – 役割またはユーザーに割り当てることができる管理権限の集まり。権利プロファイルには、承認から構成されるもの、セキュリティー属性を持つコマンドから構成されるもの、ほかの権利プロファイルから構成されるものがあります。権利プロファイルは、セキュリティー属性をグループ化する手段として便利です。
- 役割 – 特権付きアプリケーションを実行するための特殊な識別情報。この特殊な識別情報を取得できるのは、あらかじめ割り当てられたユーザーだけです。役割によって実行されるシステムではスーパーユーザーは不要です。スーパーユーザー権限はいくつかの役割に分散されます。たとえば、役割が 2 つ存在するシステムでは、セキュリティー作業をセキュリティー役割によって処理し、セキュリティー関連ではないシステム管理作業を他方の役割で処理できます。役割を細かく分割することも可能です。たとえば、暗号化フレームワーク、プリンタ、システム時刻、ファイルシステム、監査などをそれぞれ処理する個別の管理役割を作成できます。

次の図では、各 RBAC 要素の動作を示します。

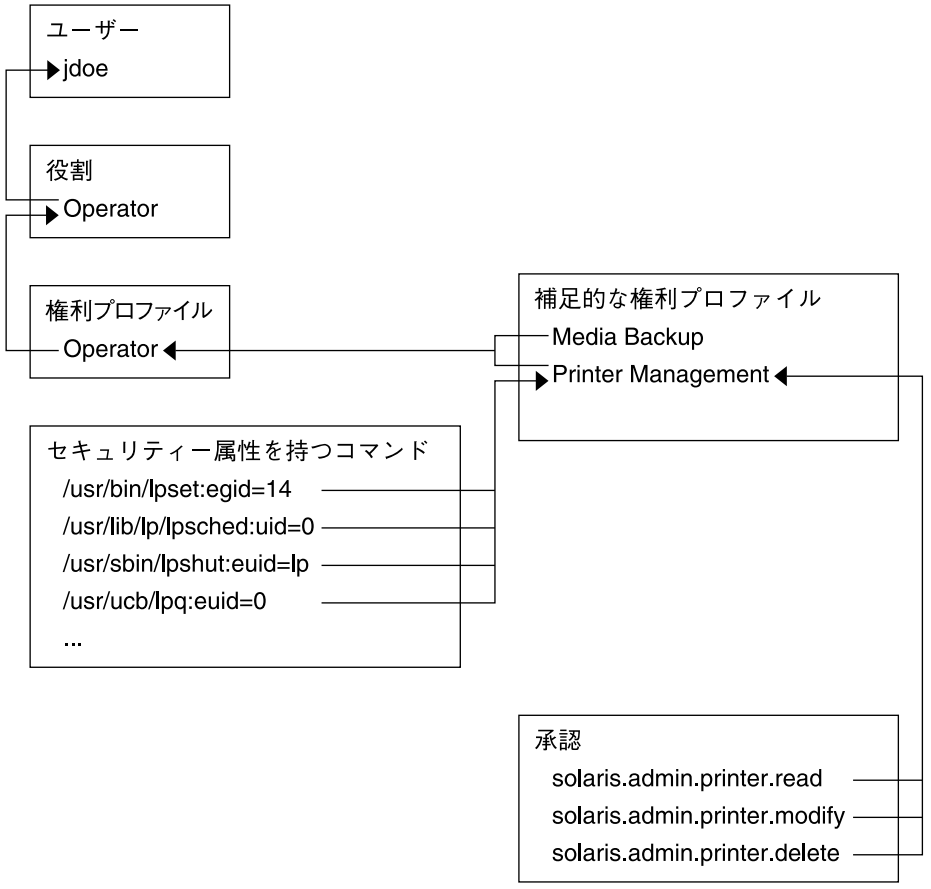
図 8–1 Oracle Solaris RBAC 要素の関係



RBAC では、ユーザーに役割が割り当てられます。役割を引き受けると、ユーザーはその役割の権限を利用できる状態となります。役割の権限は権利プロファイルから取得されます。権利プロファイルには、承認、直接割り当てられた特権、特権付きコマンド、その他の補足的な権利プロファイルを含めることができます。特権付きコマンドは、セキュリティー属性を使用して実行されるコマンドです。

次の図は、Network Security 役割と Network Security 権利プロファイルを使用して、RBAC 関係を示したものです。

図 8–2 Oracle Solaris RBAC 要素の関係の例



Network Security 役割は、IPsec、 wifi、 およびネットワークリンクを管理するために使用されます。この役割は、ユーザー jdoe に割り当てられています。jdoe は、この役割に切り替えて役割のパスワードを入力することにより、この役割を引き受けることができます。

Network Security 権利プロファイルは Network Security 役割に割り当てられています。Network Security 権利プロファイルには、Network Wifi Security、 Network Link Security、 および Network IPsec Management という、順番に評価される補助プロファイルが含まれています。これらの補助プロファイルが役割の一次タスクに追加されます。

Network Security 権利プロファイルには、直接割り当てられた 3 つの承認と、セキュリティー属性を持つ 2 つのコマンドがありますが、直接割り当てられた特権はありません。補助権利プロファイルには、直接割り当てられた承認があり、それらの 2 つにはセキュリティー属性を持つコマンドがあります。Network Security 役割で

は、jdoe はこれらのプロファイル内で割り当てられたすべての承認を持ち、これらのプロファイル内のセキュリティー属性を持つすべてのコマンドを実行できます。jdoe はネットワークセキュリティーを管理できます

特権エスカレーション

Oracle Solaris では、管理者がセキュリティーを構成するとき、高い柔軟性が提供されます。 ソフトウェアがインストールされた状態では、特権エスカレーションは許可されません。 特権エスカレーションは、意図していたよりも多くの管理権限がユーザーまたはプロセスに与えられたときに発生します。このため、特権は単なる特権ではなく、あらゆるセキュリティー属性を意味します。

Oracle Solaris ソフトウェアには、root ユーザーにのみ割り当てられるセキュリティー属性が含まれています。他のセキュリティー保護が存在する状態で、管理者は root ユーザー用に設計された属性を別のアカウントに割り当てることができますが、そのような割り当ては注意して行う必要があります。

たとえば、Media Restore という権利プロファイルが存在しますが、これは他の権利プロファイルの一部ではありません。 Media Restore はルートファイルシステム全体へのアクセスを提供するため、これを使用することで特権のエスカレーションが可能です。故意に改ざんされたファイルや交換したメディアを復元できます。デフォルトでは、この権利プロファイルを持つのは root ユーザーのみです。

特権セキュリティー属性に影響するエスカレーションについては、280 ページの「特権エスカレーションの防止」を参照してください。

RBAC の承認

「承認」は、役割またはユーザーに許可できる個別の権限です。承認は、ユーザーアプリケーションレベルでポリシーを適用します。

承認は役割またはユーザーに直接割り当てることができますが、承認を権利プロファイルに入れておくことが最善の方法です。権利プロファイルは役割に追加され、役割がユーザーに割り当てられます。例については、図 8–2 を参照してください。

RBAC に準拠したアプリケーションは、ユーザーの承認を確認してから、アプリケーションまたはアプリケーション内の特定の操作に対するアクセス権を許可します。この承認の確認は、従来の UNIX アプリケーションが行なっていた UID=0 の確認に代わるものです。承認についての詳細は次の節で説明します。

- 248 ページの「承認の命名と委託」
- 252 ページの「auth_attr データベース」
- 258 ページの「承認を必要とするコマンド」

承認と特権

特権は、カーネル内でセキュリティーポリシーを適用します。承認と特権の違いは、セキュリティーポリシーが適用されるレベルにあります。プロセスに適切な特権がないと、特権化された処理の実行がカーネルによって防止される可能性があります。適切な承認が与えられていないユーザーは、特権付きアプリケーションを使用できなかったり、特権付きアプリケーションに含まれるセキュリティーの厳しい処理を実行できなかったりする可能性があります。特権についての詳細は、199 ページの「特権 (概要)」を参照してください。

特権付きアプリケーションと RBAC

システム制御に優先するアプリケーションとコマンドは、特権付きアプリケーションとみなされます。アプリケーションは、`UID=0` のようなセキュリティー属性、特権、および承認によって特権化されます。

UID と GID を確認するアプリケーション

`root` (`UID=0`) やその他の特殊な UID または GID を確認する特権付きアプリケーションは、UNIX 環境に古くから存在します。権利プロファイルのメカニズムによって、特定の ID を必要とするコマンドを分離できます。任意のユーザーがアクセスできるコマンドの ID を変更する代わりに、実行セキュリティー属性を指定したコマンドとして権利プロファイルに配置できます。その権利プロファイルを持つユーザーまたは役割であれば、スーパーユーザー以外でもプログラムを実行できます。

ID として、実 ID または実効 ID を指定できます。実効 ID を割り当てた場合は、実 ID より優先されます。実効 ID は、ファイルアクセス権ビットの `setuid` 機能に相当します。実行 ID は、監査のために UID の識別も行います。ただし、`root` の実 UID を要求するシェルスクリプトやプログラムのために、実 ID も設定できます。たとえば、`pkgadd` コマンドは、実効 UID ではなく実 UID を要求します。コマンドを実行する上で実効 ID では十分でない場合は、ID を実 ID に変更する必要があります。手順については、234 ページの「権利プロファイルを作成または変更する方法」を参照してください。

特権を確認するアプリケーション

特権付きアプリケーションでは、特権の使用を確認できます。RBAC 権利プロファイルメカニズムを使用すれば、特定のコマンドに特権を指定できます。この方法では、アプリケーションまたはコマンドの実行時にスーパーユーザー権限を要求するのではなく、実行セキュリティー属性を指定してコマンドを権利プロファイルとして分離します。この権利プロファイルを持つユーザーまたは役割は、そのコマンドの実行に必要な特権だけを使用してコマンドを実行できます。

特権を確認するコマンドとして次のようなものがあります。

- Kerberos コマンド (kadmin、 kprop、 kdb5_util など)
- ネットワークコマンド (ifconfig、 routeadm、 snoop など)
- ファイルコマンドとファイルシステムコマンド (chmod、 chgrp、 mount など)
- プロセスを制御するコマンド (kill、 pcred、 rcapadm など)

コマンドに特権を指定して権利プロファイルに追加する方法は、234 ページの「権利プロファイルを作成または変更する方法」を参照してください。特定のプロファイルに特権が存在するか確認するコマンドを調べる場合は、270 ページの「割り当てられた特権の判断」を参照してください。

承認を確認するアプリケーション

Oracle Solaris には、承認を確認するコマンドも用意されています。当然ながら、root ユーザーにはあらゆる承認が与えられます。このため、root ユーザーは任意のアプリケーションを実行できます。承認があるかどうかを確認するアプリケーションには次のようなものがあります。

- Solaris 管理コンソールのすべてのツール
- 監査管理用のコマンド (auditconfig、 auditreduce など)
- プリンタ管理用のコマンド (lpadmin、 lpfilter など)
- バッチジョブ関連のコマンド (at、 atq、 batch、 crontab など)
- デバイス向けのコマンド (allocate、 deallocate、 list_devices、 cdrw など)

承認の確認のためにスクリプトまたはプログラムをテストする場合は、例 9–24 を参照してください。承認が必要なプログラムを作成する場合は、『Oracle Solaris セキュリティーサービス開発ガイド』の「承認について」を参照してください。

RBAC の権利プロファイル

「権利プロファイル」は、役割またはユーザーに割り当てることができるシステムの設定より優先されるオペレーションの集合です。権利プロファイルには、承認、割り当て済みのセキュリティー属性が指定されたコマンド、その他の権利プロファイルを含めることができます。権利プロファイル情報は、prof_attr データベースと exec_attr データベースに分けて保存されます。権利プロファイル名と承認は、prof_attr データベースに置かれます。権利プロファイル名と、割り当て済みのセキュリティー属性が指定されたコマンドは、exec_attr データベースに置かれます。

権利プロファイルの詳細は、次の節を参照してください。

- 243 ページの「権利プロファイルの内容」
- 254 ページの「prof_attr データベース」
- 255 ページの「exec_attr データベース」

RBAC の役割

「役割」は、特権付きアプリケーションを実行できる特別な種類のユーザーアカウントです。役割は、ユーザーアカウントと同じ方法で作成され、ホームディレクトリ、グループ割り当て、パスワードなどを持ちます。権利プロファイルと承認は、役割に管理権限を与えます。役割は、ほかの役割やユーザーから権限を継承することはできません。個々の役割によってスーパーユーザー権限が分配されるため、安全な管理が行えるようになります。

ユーザーが役割を引き受けると、その役割の属性がすべてのユーザー属性を置き換えます。役割情報は、`passwd`、`shadow`、および `user_attr` データベースに保存されます。役割情報の追加は、`audit_user` データベースに対して行うことができます。役割の設定についての詳細は、次の節を参照してください。

- 211 ページの「RBAC の実装を計画する方法」
- 217 ページの「コマンド行から役割を作成する方法」
- 232 ページの「役割のプロパティーを変更する方法」

各役割は、複数のユーザーに割り当てることができます。同じ役割になるすべてのユーザーは、同じ役割のホームディレクトリを持ち、同じ環境で動作し、同じファイルへのアクセス権を持ちます。ユーザーは、コマンド行から `su` コマンドを実行し、役割名とパスワードを指定して役割を引き受けることができます。Solaris 管理コンソールツールで役割を引き受けることもできます。

役割自体が直接ログインすることはできません。ユーザーがまずログインし、続いて役割を引き受けます。役割を引き受けたあとは、その役割を終了するまでほかの役割を引き受けることはできません。つまり、その役割を終了すれば、ほかの役割を引き受けることができます。

222 ページの「`root` ユーザーを役割にする方法」に示しているように、`root` ユーザーを役割に変更することによって、匿名の `root` ログインを防ぐことができます。プロファイルシェルコマンド `pfexec` が監査の対象となっている場合、監査トレールにはログインユーザーの実 UID、そのユーザーが引き受けた役割、およびその役割が行ったアクションが記録されます。役割による処理を行うシステムまたは特定のユーザーを監査する方法については、221 ページの「役割を監査する方法」を参照してください。

Oracle Solaris には、事前に定義された役割は用意されていません。ただし、ソフトウェアに付属している権利プロファイルは、役割に対応付けられています。たとえば、Primary Administrator 権利プロファイルを使用すると、Primary Administrator 役割を作成できます。

- Primary Administrator 役割を設定する方法については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。
- ほかの役割の設定方法については、213 ページの「GUI を使用して役割の作成および割り当てを行う方法」を参照してください。

- コマンド行で役割を作成する方法については、229 ページの「RBAC の管理(作業マップ)」を参照してください。

プロファイルシェルと RBAC

役割は、特権付きアプリケーションを Solaris 管理コンソール起動ツールから実行することも、あるいは「プロファイルシェル」から実行することもできます。プロファイルシェルプロファイルシェルは、権利プロファイルに含まれるセキュリティー属性を認識する特殊なシェルです。プロファイルシェルは、ユーザーが `su` コマンドを実行して役割を引き受けたときに起動されます。プロファイルシェルには、`pfsh`、 `pfcsh`、 および `pfksh` があります。これらはそれぞれ、Bourne シェル (`sh`)、 C シェル (`csh`)、 および Korn シェル (`ksh`) に対応します。

権利プロファイルが直接割り当てられたユーザーは、セキュリティー属性が指定されたコマンドを実行する場合、プロファイルシェルを起動する必要があります。操作性やセキュリティーに関する考慮事項については、198 ページの「セキュリティー属性を直接割り当てる場合に考慮すべきセキュリティー事項」を参照してください。

プロファイルシェルで実行されるコマンドはすべて、監査の対象に含めることができます。詳細は、221 ページの「役割を監査する方法」を参照してください。

ネームサービススコープと RBAC

ネームサービススコープは、RBAC を理解する上で重要な概念です。役割の適用範囲は、個々のホストに限定されることもあれば、ネームサービス (NIS、 NIS+、 または LDAP) のサービス対象となるすべてのホストとなることもあります。システムにおけるネームサービススコープは、`/etc/nsswitch.conf` で指定します。検索は、最初に一致した時点で停止します。たとえば、権利プロファイルが 2 つのネームサービススコープに存在する場合、最初のネームサービススコープに含まれるエントリだけが使用されます。最初に一致したものが `files` の場合、役割の適用範囲はローカルホストに限定されます。

セキュリティー属性を直接割り当てる場合に考慮すべきセキュリティー事項

一般にユーザーは、役割を通して管理権限を取得します。承認と、特権付きコマンドは、権利プロファイル内でグループ化されます。権利プロファイルは役割に含められ、役割がユーザーに割り当てられます。

権利プロファイルとセキュリティー属性を直接割り当てることも可能です。

- ユーザーには、権利プロファイル、特権、および承認を直接割り当てることができます。
- 役割とユーザーには、特権と承認を直接割り当てることができます。

しかし、特権の割り当てを直接行うことは安全とは言えません。特権が直接割り当てられたユーザーと役割は、カーネルがその特権を要求するどの場合でもセキュリティーポリシーに優先します。安全なやり方は、特権をコマンドのセキュリティー属性として権利プロファイル内で割り当てる方法です。そうすると、その特権は、その権利プロファイルを持つユーザーが、そのコマンドでのみ使用できます。

承認はユーザーレベルで作用するため、承認の直接割り当ては特権の直接割り当てよりリスクが小さいと言えます。しかし、承認が与えられることで、ユーザーは監査フラグの割り当てなどの高いセキュリティーが求められる作業も実施できるようになります。

ユーザーに直接割り当てられた権利プロファイルでは、セキュリティーよりも操作性に関して問題が多く発生します。権利プロファイルでセキュリティー属性が指定されたコマンドは、プロファイルシェルでしか実行できません。ユーザーはプロファイルシェルを開き、そのシェルにコマンドを入力する必要があります。権利プロファイルが割り当てられた役割は、プロファイルシェルを自動的に取得します。このため、コマンドはその役割のシェルで正常に実行されます。

特権 (概要)

プロセス権管理は、プロセスをコマンド、ユーザー、役割、またはシステムのいずれかのレベルに限定するために利用できます。Oracle Solaris では、特権を通してプロセス権管理を行います。特権は、システムに対するすべてのスーパーユーザー権限を 1 人のユーザーまたは 1 つのプロセスだけが持っている場合に伴うセキュリティーリスクを軽減します。特権と RBAC は、従来のスーパーユーザーモデルの代替となる強力なモデルを提供します。

- RBAC については、188 ページの「役割によるアクセス制御 (概要)」を参照してください。
- 特権の管理方法については、第 11 章「特権 (手順)」を参照してください。
- 特権に関する参照情報については、第 12 章「特権 (参照)」を参照してください。

カーネルプロセスを保護する特権

特権は、処理を実行するためにプロセスが必要とする個々の権利です。この権利はカーネルにおいて実効性があります。Oracle Solaris の特権「基本セット」の範囲内で

動作するプログラムは、システムセキュリティーポリシーの範囲内で動作します。setuid プログラムは、システムセキュリティーポリシーの範囲を超えて動作するプログラムの例です。特権を使用することで、プログラムは setuid を呼び出さなくてすみます。

特権は、システム上で行える処理を個々にエミュレートします。プログラムは、その実行に必要な最小限の特権で実行できます。たとえば、日付を設定してその日付を管理ファイルに書き込むプログラムは、file_dac_write 特権と sys_time 特権があれば実行できます。この機能を利用することで、あらゆるプログラムを root として実行する必要がなくなります。

従来、システムは特権モデルを追求してきませんでした。むしろ、システムはスーパーユーザーモデルを使用していました。スーパーユーザーモデルでは、プロセスは root またはユーザーとして実行されます。ユーザープロセスは、ユーザーのディレクトリとファイルにだけ作用するように限定されました。root プロセスは、システム上の任意の場所にディレクトリとファイルを作成できました。ユーザーのディレクトリ以外の場所にディレクトリを作成する必要があるプロセスは、UID=0 を使用して (つまり root として) 実行されました。セキュリティーポリシーは、システムファイルを保護するのに、任意アクセス制御 (Discretionary Access Control、 DAC) に依存していました。デバイスノードは、DAC によって保護されました。たとえば、グループ sys が所有しているデバイスをオープンできるのはこのグループのメンバーだけでした。

しかし、setuid プログラムやファイルアクセス権、管理アカウントなどは悪用される危険性があります。setuid プロセスに許可されているアクションは、このプロセスがその処理に必要な数を上回っています。setuid プログラムが侵入者に攻撃された場合には、全権を有する root ユーザーとしてふるまわれてしまいます。同様に、root パスワードにアクセスできるユーザーは誰でもシステム全体に損害を与えかねません。

これとは対照的に、特権によるポリシーを適用するシステムでは、ユーザー権限と root 権限との間に段階を付けることができます。たとえば、通常のユーザー権限を超える処理が可能となる特権を特定のユーザーに与えたり、現在持っている特権よりも少なくなるように root の特権数を制限したりできます。RBAC では、特権で実行されるコマンドを権利プロファイルとして分離し、これを 1 人のユーザーまたは 1 つの役割に割り当てることができます。表 8–1 は、特権を使用した RBAC モデルで利用できるユーザー権限とスーパーユーザー権限間の段階的な変化を示しています。

特権モデルでは、スーパーユーザーモデルより高いレベルのセキュリティーが実現されます。プロセスから削除された特権が悪用される可能性はありません。プロセス特権を利用することで、プログラムまたは管理アカウントが全機能のアクセス権を得ないように防止できます。プロセス特権は、DAC 保護だけでは弱点を突かれてアクセス権が取得される可能性があることから、重要なファイルを保護するための追加手段にもなります。

特権を使用することで、必要な機能しか持たないようにプログラムとプロセスを制限できます。これは、「最小特権の原則」と呼ばれます。最小特権を実装するシステムでは、プロセスを取得した侵入者がアクセスできるのはそのプロセスに与えられた特権だけであり、システムのほかの部分を攻撃することはできません。

特権の種類

特権は、それぞれの領域に基づいて論理的にグループ化されます。

- FILE 特権 – 文字列 file で始まる特権は、ファイルシステムオブジェクトに対して作用します。たとえば、file_dac_write 特権は、ファイルへの書き込みの際に任意アクセス制御に優先します。
- IPC 特権 – 文字列 ipc で始まる特権は、IPC オブジェクトアクセス制御を無効にします。たとえば、ipc_dac_read 特権を使用すると、DAC によって保護されている遠隔共有メモリーを読み取るプロセスが可能となります。
- NET 特権 – 文字列 net で始まる特権は、特定のネットワーク機能へのアクセスを可能にします。たとえば、net_rawaccess 特権を使用すると、デバイスをネットワークに接続できます。
- PROC 特権 – 文字列 proc で始まる特権は、プロセスがそれ自体の限定されたプロパティーを変更できます。PROC 特権の中には、ごくわずかな効果しかない特権もあります。たとえば、proc_clock_highres 特権は、プロセスが高分解能タイマーを使用できます。
- SYS 特権 – 文字列 sys で始まる特権は、各種のシステムプロパティーに対する無制限のアクセス権をプロセスに与えます。たとえば、sys_linkdir 特権を使用すると、プロセスはディレクトリに対するハードリンクの確立と解除が行えます。

特権の中にはシステムに対する影響が少ないものもあれば、大きな影響を与えるものもあります。次の proc_taskid 特権の定義は、この特権の影響が小さいことを示しています。

```
proc_taskid
        Allows a process to assign a new task ID to the calling process.
```

file_setid 特権の定義は、この特権の影響が大きいことを示しています。

```
net_rawaccess
        Allow a process to have direct access to the network layer.
```

各特権の説明は、privileges(5) のマニュアルページを参照してください。コマンド ppriv -lv を実行すると、各特権の説明が標準出力に送られます。

特権を使用したシステムにおける管理上の相違点

特権を持つシステムと特権を持たないシステムとでは、明白な違いがいくつかあります。次の表に相違点の一部を示します。

表 8–2 特権を持つシステムと特権を持たないシステムとの明白な違い

機能	特権なし	特権
デーモン	デーモンが root として実行されます。	デーモンが、ユーザー daemon として実行されます。 たとえば、デーモン lockd、 nfsd、 rpcbind には適切な特権が割り当てられており、daemon として実行されます。
ログファイルの所有権	ログファイルは root によって所有されます。	ログファイルはログファイルを作成した daemon によって所有されます。root ユーザーがこのファイルを所有することはありません。
エラーメッセージ	エラーメッセージでスーパーユーザーが言及されます。 たとえば、chroot: not superuser。	エラーメッセージで特権の使用が言及されます。 たとえば、chroot エラーと同等のエラーメッセージは chroot: exec failed。
setuid プログラム	プログラムは、通常のユーザーに許可されていない作業を行うのに setuid を使用します。	多くの setuid プログラムは、特権を使用して実行されるように変更されました。 たとえば、ユーティリティー ufsdump、 ufsrestore、 rsh、 rlogin、 rcp、 rdist、 ping、 traceroute、 newtask は特権を使用します。
ファイルのアクセス権	デバイスアクセス権は DAC によって制御されます。たとえば、グループ sys のメンバーは /dev/ip を開くことができます。	デバイスを開くことができるユーザーをファイルアクセス権 (DAC) が予測することはありません。デバイスは、DAC とデバイスポリシーによって保護されます。 たとえば、/dev/ip ファイルには 666 アクセス権がありますが、デバイスを開くことができるのは適切な特権を持つプロセスだけです。raw ソケットは依然として DAC によって保護されます。
監査イベント	su コマンドの使用の監査によって、多くの管理機能がカバーされます。	特権の使用の監査によって、ほとんどの管理機能がカバーされます。pm や as の監査クラスには、デバイスポリシーを設定する監査イベントや特権を設定する監査イベントが含まれます。
プロセス数	プロセスは、プロセスの所有者によって保護されます。	プロセスは特権によって保護されます。プロセス特権とプロセスフラグは、/proc/<pid> ディレクトリ内の新しいエントリである priv として確認できます。

表 8–2 特権を持つシステムと特権を持たないシステムとの明白な違い (続き)

機能	特権なし	特権
デバッグ	コアダンプ内で特権の言及はありません。	コアダンプの ELF 注記セクションで、NT_PRPRIV および NT_PRPRIVINFO の注記としてプロセス特権とフラグについての情報が示されます。 ppriv などのユーティリティーによって、適切にサイズ設定された特権セットの適切な数が示されます。これらのユーティリティーは、ビットセット内のビットを正しく特権名にマッピングします。

特権とシステム資源

Solaris 10 8/07 リリースから、PRIV_PROC_LOCK_MEMORY 特権が割り当てられたプロセスのメモリー消費を、project.max-locked-memory および zone.max-locked-memory 資源制御を使って制限できるようになりました。プロセスはこの特権を使うことで、物理メモリー内のページをロックできます。

PRIV_PROC_LOCK_MEMORY 特権を権利プロファイルに割り当てると、この特権を持つプロセスに、すべてのメモリーをロックする権限を与えることになります。安全対策として、この特権を持つユーザーがすべてのメモリーをロックできないように、資源制御を設定してください。特権付きプロセスが非大域ゾーン内で実行される場合には、zone.max-locked-memory 資源制御を設定します。特権付きプロセスがシステム上で実行される場合には、プロジェクトを作成し、project.max-locked-memory 資源制御を設定します。これらの資源制御については、『Oracle Solaris のシステム管理 (Oracle Solaris コンテナ : 資源管理と Oracle Solaris ゾーン)』の第 6 章「資源制御 (概要)」および『Oracle Solaris のシステム管理 (Oracle Solaris コンテナ : 資源管理と Oracle Solaris ゾーン)』の第 17 章「非大域ゾーンの構成 (概要)」を参照してください。

特権の実装方法

各プロセスには、プロセスが特定の特権を使用できるかどうかを判断する 4 セットの特権があります。カーネルは、特権「有効セット」を自動的に計算します。初期の特権「継承可能セット」は変更できます。特権を使用するように作成されているプログラムは、そのプログラムで使用する特権の「許可されたセット」を減らすことができます。特権「制限セット」は縮小できます。

- 有効特権セット **(E)** – 現在有効である特権の集合です。プロセスは、許可されたセット内の特権を有効セットに追加できます。プロセスは、E から特権を削除することもできます。
- 許可された特権セット **(P)** – 使用できる特権の集合です。プログラムは、継承または割り当てを通して特権を使用できます。実行プロファイルは、プログラムに特権を割り当てる方法の 1 つです。setuid コマンドは、root が持つすべての特権を

プログラムに割り当てます。許可されたセットから特権を削除することはできますが、許可されたセットに特権を追加することはできません。P から削除された特権は、E からも自動的に削除されます。

「特権を認識する」プログラムは、そのプログラムがまったく使用することのない特権をそのプログラムの許可されたセットから削除します。この方法では、不要な特権がそのプログラムや悪質なプロセスによって悪用されることが防止されます。特権を認識するプログラムの詳細は、『Oracle Solaris セキュリティーサービス開発ガイド』の第 2 章「特権付きアプリケーションの開発」を参照してください。

- 継承可能な特権セット **(I)** – exec 呼び出しでプロセスが継承できる特権の集合です。exec 呼び出しのあと、setuid プログラムの特殊なケースを除き、許可されたセットと有効セットは同じになります。

  setuid プログラムの場合、exec 呼び出しのあと、継承可能セットがまず制限セットによって制限されます。続いて、継承された特権のセット (I) から制限セット (L) が除かれたものが、そのプロセスの P と E に割り当てられます。

- 制限特権セット **(L)** – プロセスおよびその子プロセスで使用できる特権の外側の限界です。デフォルトでは、制限セットはすべての特権です。プロセスは制限セットを縮小することはできますが、制限セットを拡張することはできません。L は I の制限に使用されます。このため、L は exec の時点で P と E を制限します。

  特権が割り当てられたプログラムを含むプロファイルがユーザーに割り当てられている場合、通常そのユーザーはそのプログラムを実行できます。未変更のシステムでは、プログラムの割り当て済み特権はユーザーの制限セットの範囲内です。プログラムに割り当てられている特権は、ユーザーの許可されたセットの一部になります。特権を割り当てられたプログラムを実行するには、プロファイルシェルからそのプログラムを実行する必要があります。

カーネルは、「基本特権セット」を認識します。変更されていないシステムの場合、各ユーザーの初期の継承可能セットはログイン時の基本セットと同じです。ユーザーの初期の継承可能セットは変更が可能です。基本セットは変更できません。

未変更のシステムでは、ログイン時のユーザーの特権セットは次のようになります。

```
E (Effective): basic
I (Inheritable): basic
P (Permitted): basic
L (Limit): all
```

このため、ログイン時には各ユーザーの基本セットは、それぞれの継承可能セット、許可されたセット、および有効セットに含まれます。ユーザーの制限セットにはすべての特権が含まれます。ユーザーの有効セットに特権を追加するには、そのユーザーに権利プロファイルを割り当てる必要があります。権利プロファイルは、通常、特権が追加されたコマンドを含みます。特権はユーザーまたは

役割に直接割り当てることもできますが、そのような特権割り当てはリスクを伴います。リスクの詳細は、198 ページの「セキュリティー属性を直接割り当てる場合に考慮すべきセキュリティー事項」を参照してください。

プロセスが特権を取得する方法

プロセスは特権を継承できます。あるいは、プロセスに特権を割り当てることもできます。プロセスは、その親から特権を継承します。ログイン時に、ユーザーの初期継承可能特権セットによって、そのユーザーのプロセスで使用できる特権が決まります。ユーザーの当初のログインの子プロセスはすべて、このセットを継承します。

プログラム、ユーザー、および役割に特権を直接割り当てることもできます。プログラムで特権が必要な場合は、権利プロファイル内でそのプログラムの実行可能ファイルに特権を割り当てます。そのプログラムの実行を許可されたユーザーまたは役割には、そのプログラムが入ったプロファイルを割り当てます。ログイン時、あるいはプロファイルシェルが開かれている場合、プログラムの実行可能ファイルがプロファイルシェルで入力されると、そのプログラムは特権を使用して実行されます。たとえば、Object Access Management プロファイルが含まれる役割は、`file_chown` 特権を使用して `chmod` コマンドを実行できます。

付加的な特権が直接割り当てられたプログラムを役割またはユーザーが実行する場合、割り当てられているその特権は役割またはユーザーの継承可能セットに追加されます。特権が割り当てられたプログラムの子プロセスは、親プロセスの特権を継承します。子プロセスが親プロセスよりも多くの特権を必要とする場合には、子プロセスに直接それらの特権を割り当てる必要があります。

特権を使用するように作成されているプログラムは、「特権を認識するプログラム」と呼ばれます。特権を認識するプログラムは、その実行中に特権の使用を有効にしたり無効にしたりします。本番環境で使用するためには、この特権をそのプログラムに割り当てる必要があります。

特権を認識するコードの例は、『Oracle Solaris セキュリティーサービス開発ガイド』の第 2 章「特権付きアプリケーションの開発」を参照してください。特権を必要とするプログラムに特権を割り当てる場合は、266 ページの「特権をコマンドに追加する方法」 を参照してください。

特権の割り当て

特権の割り当ては、システム管理者の権限を与えられたユーザーによって行われます。この管理者は、通常、権利プロファイル内でコマンドに特権を割り当てます。この割り当てのあと、この権利プロファイルを役割またはユーザーに割り当てます。Solaris 管理コンソールには、特権を割り当てるためのグラフィカル

ユーザーインタフェース (GUI) が用意されています。特権の割り当ては、smuser や smrole のようなコマンドを使用しても行えます。GUI を使用して特権を割り当てる方法の詳細は、第 9 章「役割によるアクセス制御の使用 (手順)」を参照してください。

特権をユーザーに直接割り当てることもできます。セッションで特権を適切に使用すると信頼できるユーザーには、特権を直接割り当てることができます。直接の割り当てが適するものとしては、影響の少ない特権 (proc_clock_highres など) が挙げられます。影響が広範囲におよぶ特権 (file_dac_write など) は、直接の割り当てに適しません。

ユーザーまたはシステムに対して特権を拒否することもできます。ユーザーまたはシステムの初期継承可能セットまたは制限セットから特権を削除する場合は、注意が必要です。

ユーザーまたは役割の特権の拡張

ユーザーと役割は、継承可能特権セットと制限特権セットを持ちます。制限セットには初めにすべての特権が設定されるため、このセットは拡張できません。初期継承可能セットは、ユーザー、役割、システムを対象に拡張できます。プロセスには、継承可能セットに含まれない特権も割り当てることができます。

特権を追加するのにもっとも適した方法は、プロセスごとの特権割り当てです。ユーザーが実行できる特権化された処理の数は、役割の引き受けをそのユーザーに許可することで増やせます。役割には、追加された特権が指定されたコマンドを含むプロファイルを割り当てます。役割を引き受ける際に、ユーザーはその役割のプロファイルシェルを取得します。役割のプロファイルに指定されたコマンドをその役割のシェルで入力すると、追加された特権を使用してコマンドが実行されます。

プロファイルは、ユーザーが引き受ける役割に対してではなく、ユーザーに割り当てることもできます。プロファイルは、追加された特権が指定されたコマンドを含みます。プロファイルシェル (pfksh など) を開く場合、ユーザーは自分のプロファイル内のコマンドを特権を使用して実行できます。通常のシェルでは、特権によるコマンドの実行は行われません。特権が指定されたプロセスは、特権化されたシェルでしか実行できません。

ユーザー、役割、またはシステムの初期継承可能特権セットを拡張するのは、特権を割り当てる方法として安全とは言えません。継承可能セット内の特権はすべて、許可されたセットと有効セット内に存在します。シェル内でユーザーまたは役割が入力するコマンドはすべて、直接割り当てられた特権を使用できます。直接割り当てられた特権を使用すると、ユーザーまたは役割は自分の管理責務の範囲を超えた処理を簡単に行えます。

システムの初期継承可能特権セットを拡張すると、そのシステムにログインするユーザー全員の基本特権が拡張されます。このような直接割り当てを行うと、通常のユーザーには禁止されている処理でもシステムの全ユーザーが簡単に行えるようになります。

注 – 制限セットには初めにすべての特権が設定されるため、このセットは拡張できません。

ユーザーまたは役割の特権の制限

特権を削除することで、ユーザーまたは役割が特定の作業を行わないようにすることができます。特権は、初期継承可能セットから削除することも、制限セットから削除することもできます。デフォルトセットよりも小さい初期継承可能セットまたは制限セットを配布する場合は、あらかじめ特権の削除を慎重にテストすることが望まれます。たとえば、初期継承可能セットから特権を削除したためにユーザーがログインできなくなる可能性があります。また、制限セットから特権を削除したために古い `setuid` プログラムでその特権が使えなくなり、プログラムが失敗する可能性があります。

スクリプトへの特権の割り当て

スクリプトは、コマンドと同様に実行が可能です。このため、コマンドに特権を追加する場合と同じ方法で、権利プロファイルでスクリプトに特権を追加できます。スクリプトは、プロファイルを割り当てられたユーザーまたは役割がプロファイルシェル内でそのスクリプトを実行する場合に、追加された特権を使用して実行されます。特権を必要とするコマンドがスクリプトに含まれる場合は、その特権を指定したコマンドもプロファイルに含める必要があります。

特権を認識するプログラムは、プロセス単位で特権を制限できます。特権を認識するプログラムを使用するジョブは、プログラムで必要な特権だけを実行可能ファイルに割り当てます。続いて管理者はこのプログラムのテストを行い、作業が正常に行われるか確認します。また、プログラムが特権を悪用しないかも確認します。

特権とデバイス

スーパーユーザーモデルではファイルアクセス権だけでシステムインタフェースが保護されていますが、特権モデルでは特権を使用してシステムインタフェースを保護します。特権を使用したシステムでは、インタフェースを保護するほどの強さはファイルアクセス権にありません。`proc_owner` などの特権は、ファイルアクセス権を無効にした上でファイルシステム全体へのフルアクセス権を与える可能性があります。

このため、デバイスを開くにはデバイスディレクトリの所有権では不十分です。たとえば、/dev/ip デバイスを開く許可がグループ sys のメンバーに自動的に与えられることはありません。/dev/ip のファイルアクセス権は 0666 ですが、デバイスを開くには net_rawaccess 特権が必要です。

デバイスポリシーは特権によって制御されます。getdevpolicy コマンドは、各デバイスのデバイスポリシーを表示します。デバイス構成コマンド devfsadm は、デバイスポリシーをインストールします。devfsadm コマンドは、デバイスの読み取りまたは書き込みを行うため open に特権セットを設定します。詳細は、getdevpolicy(1M) および devfsadm(1M) のマニュアルページを参照してください。

デバイスポリシーを使えば、デバイスを開く権限をより柔軟に付与できます。デフォルトのデバイスポリシーとは異なる特権や、デフォルトのデバイスポリシーよりも多くの特権を要求することができます。特権要件は、デバイスポリシーに合わせて変更することも、ドライバ本体に合わせて変更することもできます。特権の変更は、デバイスドライバのインストール時、追加時、または更新時に行えます。

デバイスポリシーエントリとドライバ固有の特権を変更するには、add_drv コマンドと update_drv コマンドを使用できます。デバイスポリシーを変更するには、すべての特権を使用してプロセスを実行していなければなりません。詳細は、add_drv(1M) および update_drv(1M) のマニュアルページを参照してください。

特権とデバッグ

Oracle Solaris には、特権のエラーを修正するツールが用意されています。ppriv コマンドと truss コマンドを使用して、デバッグ結果を出力できます。この例は、ppriv(1) のマニュアルページを参照してください。操作の手順は、264 ページの「プログラムが必要とする特権を判断する方法」を参照してください。

◆ ◆ ◆ 第 9 章

9 役割によるアクセス制御の使用 (手順)

この章では、個別の役割を使用することによってスーパーユーザーの機能を分散するための作業手順について説明します。役割が使用できるメカニズムとして、権利プロファイル、承認、および特権があります。この章では、次の作業マップについて説明します。

- 209 ページの「RBAC の使用 (作業マップ)」
- 210 ページの「RBAC の構成 (作業マップ)」
- 225 ページの「役割の使用 (作業マップ)」
- 229 ページの「RBAC の管理(作業マップ)」

RBAC の概要については、188 ページの「役割によるアクセス制御 (概要)」を参照してください。第 10 章「役割によるアクセス制御 (参照)」にも参考情報が挙げられています。RBAC を使用して、または RBAC を使用しないで特権を使用する方法については、第 11 章「特権 (手順)」を参照してください。

RBAC の使用 (作業マップ)

RBAC を使用するにあたって、RBAC の計画と構成、および役割を引き受ける方法を知ることが必要です。役割について熟知すると、RBAC をさらにカスタマイズして新しい操作を処理することができます。次の作業マップにこれらの主要な作業を示します。

作業	説明	参照先
RBAC の計画と構成を行います	使用するサイトで RBAC を構成します。	210 ページの「RBAC の構成 (作業マップ)」
役割を使用します	コマンド行および Solaris 管理コンソールの GUI で役割を引き受けます。	225 ページの「役割の使用 (作業マップ)」

作業	説明	参照先
RBAC をカスタマイズします	使用するサイト用に RBAC をカスタマイズします。	229 ページの「RBAC の管理(作業マップ)」

RBAC の構成 (作業マップ)

RBAC を効果的に使用するには、計画が必要です。次の作業マップを使用して、サイトでの RBAC の計画と初期実装を行います。

作業	説明	参照先
1. RBAC を計画します	サイトのセキュリティー要件の調査、およびサイトでの RBAC の使用方法の決定が含まれます。	211 ページの「RBAC の実装を計画する方法」
2. Solaris 管理コンソールの使用方法を理解します	Solaris 管理コンソールについて熟知することが含まれます。	『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」
3. 最初のユーザーと役割を構成します	Solaris 管理コンソールで RBAC 構成ツールを使用してユーザーと役割を作成し、作成した役割を作成したユーザーに割り当てます。	『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」
4. (省略可能) 役割を引き受けることができるほかのユーザーを作成します	管理役割を引き受けることができるユーザーが確実に存在するようにします。	『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」
5. (推奨) ほかの役割を作成し、ユーザーに割り当てます	RBAC ツールを使用して特定の管理領域に対する役割を作成し、作成した役割をユーザーに割り当てます。	213 ページの「GUI を使用して役割の作成および割り当てを行う方法」
		例 9–5
	コマンド行を使用して役割を作成し、作成した役割をユーザーに割り当てます	217 ページの「コマンド行から役割を作成する方法」
		220 ページの「役割をローカルユーザーに割り当てる方法」
6. (推奨) 役割の動作を監査します	役割の動作を記録する監査イベントを含む監査クラスを事前に選択します。	221 ページの「役割を監査する方法」
7. (省略可能) root ユーザーを役割にします	セキュリティーホールである、匿名の root ログインを防ぎます。	222 ページの「root ユーザーを役割にする方法」

RBAC の構成

RBAC は、次のユーティリティーで構成することができます。

- **Solaris** 管理コンソールの **GUI** – RBAC に関連する作業は、GUI を使用して実行することをお勧めします。RBAC 要素を管理するコンソールツールは、「ユーザーツールコレクション (User Tool Collection)」にあります。
- **Solaris** 管理コンソールのコマンド – `smrole` など、Solaris 管理コンソールのコマンド行インタフェースで、ネームサービスを操作することができます。Solaris 管理コンソールコマンドを使用してサーバーに接続するときは、認証が要求されます。このため、Solaris 管理コンソールコマンドは、スクリプトでは使用できません。
- ローカルコマンド – `useradd` など、コマンド行インタフェースの `user*` と `role*` の組み合わせによって、ローカルファイルのみを操作することができます。ローカルファイルを操作するコマンドは、スーパーユーザーまたは該当する特権を持つ役割によって実行される必要があります。

▼ RBAC の実装を計画する方法

RBAC は、組織の情報リソースを管理するときに、重要な役割を果たします。RBAC を計画する際には、RBAC の機能と組織のセキュリティー要件を十分に理解しておく必要があります。

1 **RBAC の基本概念を理解します。**

188 ページの「役割によるアクセス制御 (概要)」を参照してください。RBAC を使用したシステム管理は、従来の UNIX の管理方法を使用した場合と大きく異なります。実装を開始する前に、RBAC の概念を理解する必要があります。詳細については、第 10 章「役割によるアクセス制御 (参照)」を参照してください。

2 **セキュリティーポリシーを調査します。**

組織のセキュリティーポリシーには、システムに対する潜在的な脅威を詳細に記述し、各脅威の危険性の分析結果に応じて適切な対応策を定義する必要があります。RBAC を使用したセキュリティー関連の作業とは切り離して行うことをお勧めします。推奨される役割とその構成をそのままインストールすることもできますが、セキュリティーポリシーによっては RBAC の構成のカスタマイズが必要になることがあります。

3 **組織に必要な RBAC を決定します。**

組織のセキュリティー要件に応じて、さまざまなレベルの RBAC を使用できます。

- 「RBAC を使用しない」 – すべての作業を **root** ユーザーとして実行できます。この構成では、通常のユーザーとしてログインします。その後、Solaris 管理コンソールツールを選択するときに、ユーザーとして root を入力します。
- 「役割を 1 つだけ使用する」 – この方式では、役割を 1 つ追加します。追加された 1 つの役割には、Primary Administrator 権利プロファイルが割り当てられます。この方式は、役割がスーパーユーザー機能を持つという点で、スーパーユーザーモデルと似ています。ただし、この方式では、役割を引き受けたユーザーを追跡することができます。
- 「推奨される役割」 – この方式では、Primary Administrator、 System Administrator、 および Operator の権利プロファイルに基づいた 3 つの役割が作成されます。さまざまな責任レベルの管理者がいる組織の場合は、この方式が適しています。
- 「カスタム役割」 – 独自の役割を作成して、組織のセキュリティー要件を満たすことができます。新しい役割は、既存またはカスタマイズした権利プロファイルに基づいて作成できます。責務の分離を適用するように権利プロファイルをカスタマイズする方法については、『Oracle Solaris Trusted Extensions 構成ガイド』の「Trusted Extensions での役割とユーザーの作成」を参照してください。
- 「root ユーザーを役割にする」 – この方式では、どのユーザーも **root** としてログインできないようにします。代わりに、通常のユーザーとしてログインしてから、root 役割を引き受ける必要があります。詳細については、222 ページの「root ユーザーを役割にする方法」を参照してください。

4 組織に適した推奨される役割を決定します。

推奨される役割とそのデフォルトの権利プロファイルの機能を確認します。デフォルトの権利プロファイルにより、管理者は単一のプロファイルを使用して推奨される役割を構成することができます。

推奨される役割を構成するときは、次の 3 つのデフォルトの権利プロファイルを使用できます。

- 「Primary Administrator」権利プロファイル – すべての管理タスクを実行できる役割用。ほかのユーザーに権限を与えたり、管理役割に関連付けられた権限を編集したりすることができます。この役割のユーザーは、この役割をほかのユーザーに割り当てたり、ほかのユーザーに権利を与えたりすることができます。
- 「System Administrator」権利プロファイル – セキュリティーに関係しないほとんどの管理タスクを実行できる役割用。たとえば、System Administrator は、新しいユーザーアカウントは追加できますが、パスワードを設定したりほかのユーザーに権利を与えたりすることはできません。
- 「Operator」権利プロファイル – メディアバックアップやプリンタ管理など、単純な管理タスクを実行できる役割用。

権利プロファイルの詳細については、次のいずれかを参照してください。

- /etc/security ディレクトリで、prof_attr データベースおよび exec_attr データベースの内容を参照してください。
- Solaris 管理コンソールで、権限ツールを使用して権利プロファイルの内容を表示してください。
- このドキュメントで、一般的な権利プロファイルを要約した243 ページの「権利プロファイルの内容」を参照してください。

5 追加する任意の役割または権利プロファイルが組織に適切であるかどうかを判断します。

使用するサイトで、アクセスを制限する必要があるアプリケーションを調べます。セキュリティーに影響するアプリケーション、サービス拒否が発生する可能性のあるアプリケーション、特別な管理者教育を必要とするアプリケーションには、RBAC を適用することをお勧めします。役割や権利プロファイルをカスタマイズして、組織のセキュリティー要件に対応することができます。

a. 新しい操作に必要なコマンドを決定します。

b. この操作に適切な権利プロファイルを決定します。

既存の権利プロファイルがこの操作に割り当てられていないか、または別の権利プロファイルを作成する必要がないかどうかを確認します。

c. この権利プロファイルに適した役割を決定します。

この操作の権利プロファイルを既存の役割に割り当てるかどうか、または新しい役割を作成するかどうかを決定します。既存の役割を使用する場合は、この役割を割り当てるユーザーにほかの権利プロファイルが適していないかどうかを確認します。

6 役割に割り当てるユーザーを決定します。

必要な権限だけを割り当てるために、ユーザーの信頼レベルに応じて役割を割り当てます。実行する必要のない操作にユーザーがアクセスできないようにすると、問題が発生する可能性が減少します。

▼ GUI を使用して役割の作成および割り当てを行う方法

新しい役割を作成するには、スーパーユーザーになるか、 Primary Administrator 役割を使用します。この手順では、新しい役割の作成者が Primary Administrator 役割を引き受けています。

始める前に
- 使用するサイトで役割を引き受けることができるユーザーは、すでに作成されています。ユーザーをまだ作成していない場合は、『Solarisのシステム管理(基本編)』の「Solaris管理ツールをRBACと組み合わせて使用する(作業マップ)」に記載されている手順に従ってユーザーを作成してください。
- Primary Administrator役割は、『Solarisのシステム管理(基本編)』の「Solaris管理ツールをRBACと組み合わせて使用する(作業マップ)」の手順に従って割り当て済みです。

1 **Solaris管理コンソールを起動します。**

```
# /usr/sbin/smc &
```

ログインの手順については、228ページの「Solaris管理コンソールで役割を引き受ける方法」を参照してください。

2 「管理役割(Administrative Roles)」アイコンをクリックします。

3 「アクション(Action)」メニューから「管理役割を追加(Add Administrative)」を選択します。

4 一連のダイアログボックスのフィールドに入力して新しい役割を作成します。
作成可能な役割については、例9–1から例9–4を参照してください。

ヒント–Solaris管理コンソールの各ツールは、ページの下部またはウィンドウパネル左側に情報を表示します。このインタフェースでの作業について情報が必要な場合は、いつでも「ヘルプ(Help)」を選択できます。

---

5 役割をユーザーに割り当てます。

ヒント–役割のプロパティーを入力すると、最後のダイアログボックスで、その役割のユーザーを入力するよう促されます。

---

6 端末ウィンドウで、ネームサービスキャッシュデーモンを再起動します。

```
# svcadm restart system/name-service-cache
```

詳細は、svcadm(1M)およびnscd(1M)のマニュアルページを参照してください。

例9–1 System Administrator権利プロファイルの役割の作成

この例では、セキュリティーに関係しないシステム管理タスクを新しい役割が行うことができます。役割は、次のパラメータで前述の手順を実行すると作成されます。

- 役割名: sysadmin
- 役割の完全名: System Administrator

- 役割の説明: セキュリティーに関係しない管理タスクを行う
- 権利プロファイル: `System Administrator`

  この権利プロファイルは、役割に含まれるプロファイルのリストの最上部にあります。

例 9–2 Operator 権利プロファイルの役割の作成

Operator 権利プロファイルは、プリンタを管理したりオフラインメディアにシステムをバックアップしたりすることができます。役割を交代で 1 人のユーザーに割り当てたいという場合があります。そのような場合には、「手順 1: 役割名を入力します (Step 1: Enter a Role Name) 」ダイアログボックスで、 役割メーリングリストオプションを選択します。役割は、次のパラメータで前述の手順を実行すると作成されます。

- 役割名: `operadm`
- 役割の完全名: `Operator`
- 役割の説明: 操作をバックアップする
- 権利プロファイル: `Operator`

  この権利プロファイルは、役割に含めるプロファイルのリストの先頭に配置する必要があります。

例 9–3 セキュリティー関連の権利プロファイルの役割の作成

デフォルトでは、セキュリティー関連のコマンドと権利を含む権利プロファイルだけが、Primary Administrator プロファイルとなります。Primary Administrator ほどの権限はないが、セキュリティー関連のタスクを処理できる役割を作成する場合には、役割を作成する必要があります。

次の例で、役割はデバイスを保護します。役割は、次のパラメータで前述の手順を実行すると作成されます。

- 役割名: `devicesec`
- 役割の完全名: `Device Security`
- 役割の説明: デバイスを構成する
- 権利プロファイル: `Device Security`

次の例で、役割はネットワーク上のシステムとホストのセキュリティーを保護します。役割は、次のパラメータで前述の手順を実行すると作成されます。

- 役割名: `netsec`
- 役割の完全名: `Network Security`
- 役割の説明: `IPsec`、 `IKE`、 および `SSH` を処理する
- 権利プロファイル: `Network Security`

例 9–4 適用範囲が制限された権利プロファイルの役割の作成

多くの権利プロファイルでは、適用範囲に制限があります。この例では、役割の唯一のタスクはDHCPを管理することです。役割は、次のパラメータで前述の手順を実行すると作成されます。

- 役割名: `dhcpmgt`
- 役割の完全名: `DHCP Management`
- 役割の説明: `DHCP (Dynamic Host Config Protocol)` を管理する
- 権利プロファイル: `DHCP Management`

例 9–5 ユーザーの役割の割り当ての変更

この例では、既存のユーザーに役割が追加されます。ユーザーの役割の割り当てを変更するには、Solaris管理コンソールの「ユーザー」ツールの「ユーザーアカウント(User Accounts)」アイコンをクリックし、ユーザーをダブルクリックし、オンラインヘルプに従ってユーザーの機能に役割を追加します。

注意事項 役割が持つべき機能を持っていない場合、次の内容を確認します。

- 役割の権利プロファイルが、GUIで最も権限のあるものから最も権限のないものへとリストされていること。

  たとえば、`All` 権利プロファイルがリストの最上部にある場合、セキュリティー属性で実行されるコマンドはありません。セキュリティー属性を指定したコマンドを含むプロファイルは、リストで `All` 権利プロファイルの前に来なければなりません。
- 役割の権利プロファイルのコマンドに適切なセキュリティー属性を指定していること。

  たとえば、ポリシーが `suser` のとき、 `euid=0` ではなく、`uid=0` となるコマンドもあります。
- 権利プロファイルが適切なネームサービスの適用範囲で定義されていること。権利プロファイルが定義されているネームサービスの適用範囲で役割が動作していること。
- ネームサービスのキャッシュ、`svc:/system/name-service-cache` を再起動していること。

  `nscd` デーモンには長い有効期間を設定することができます。デーモンを再起動して、現在のデータでネームサービスを更新します。

▼ コマンド行から役割を作成する方法

Solaris 管理コンソールの GUI は、RBAC を管理するための望ましい方法です。GUI の使用については、213 ページの「GUI を使用して役割の作成および割り当てを行う方法」を参照してください。この手順で説明するように、コマンド行インタフェースを使用することもできます。

注 – コマンド行インタフェースとグラフィカルユーザーインタフェースを同時に使って、RBAC を管理しないでください。構成に対して矛盾した変更が加えられ、予測していない動作が生じることがあります。両方のツールとも RBAC を管理することができますが、両方を同時に使用することはできません。

始める前に　役割を作成するには、Primary Administrator 権利プロファイルを含む役割を引き受けるか、`root` ユーザーに切り替える必要があります。

1. **Primary Administrator 役割を引き受けるか、スーパーユーザーになります。**

   Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2. **次のコマンドのいずれかを選択して、コマンド行で役割を作成します。**

   - ローカルのネームサービスの適用範囲の役割には、**`roleadd`** コマンドを使用します。

     注 – `roleadd` コマンドは、Solaris 管理コンソールの GUI やコマンド行インタフェースに比べて制限されています。`roleadd` コマンドの実行後、`usermod` コマンドを実行して役割をユーザーに割り当てる必要があります。その後、ユーザーは、220 ページの「役割をローカルユーザーに割り当てる方法」で説明するように、役割のパスワードを設定する必要があります。

```
# roleadd -c comment \
-g group -m homedir -u UID -s shell \
-P profile rolename
```

`-c` *comment*	*rolename* について説明するコメントです。
`-g` *group*	*rolename* に対するグループの割り当てです。
`-m` *homedir*	*rolename* のホームディレクトリへのパスです。
`-u` *UID*	*rolename* の UID です。
`-s` *shell*	*rolename* のログインシェルです。このシェルはプロファイルシェルである必要があります。

`-P` *profile*　　*rolename* の 1 つまたは複数の権利プロファイルです。

rolename　　新しいローカル役割の名前です。

- **`smrole add`** コマンドを使用します。

  このコマンドは、NIS、 NIS+、 LDAP などの分散ネームサービスで役割を作成します。Solaris 管理コンソールサーバーのクライアントとして動作します。

  ```
  $ /usr/sadm/bin/smrole -D domain-name \
  -r admin-role -l <Type admin-role password> \
  add -- -n rolename -a rolename -d directory\
  -F full-description -p profile
  ```

  `-D` *domain-name*　　管理対象のドメインの名前です。

  `-r` *admin-role*　　役割を変更できる管理役割の名前です。管理役割は `solaris.role.assign` 承認を得る必要があります。引き受けた役割を変更する場合、役割は `solaris.role.delegate` 承認を得る必要があります。

  `-l`　　*admin-role* のパスワードに対するプロンプトです。

  `--`　　認証オプションとサブコマンドオプションの間に必要な区切り文字です。

  `-n` *rolename*　　新しい役割の名前です。

  `-c` *comment*　　役割の機能を説明するコメントです。

  `-a` *username*　　*rolename* を引き受けることができるユーザーの名前です。

  `-d` *directory*　　*rolename* のホームディレクトリです。

  `-F` *full-description*　　*rolename* の詳細です。この説明は Solaris 管理コンソールの GUI に表示されます。

  `-p` *profile*　　*rolename* の機能に含まれる権利プロファイルです。このオプションには、役割に対する管理機能を備えたコマンドが用意されています。複数の `-p` *profile* オプションを指定することができます。

3. 変更を有効にするには、**220** ページの「役割をローカルユーザーに割り当てる方法」を参照してください。

例 9–6 smrole コマンドを使用してカスタムの Operator 役割を作成する

`smrole` コマンドは、ネームサービスで新しい役割とその属性を指定します。次の例では、 Primary Administrator が Media Backup 役割の新しいバージョンを作成しま

す。役割には、標準的なMedia Backup権利プロファイルのほか、FTP Management権利プロファイルが含まれます。コマンドによって、新しい役割のパスワードを入力するよう促されます。

```
% su - primaryadm
Password:      <Type primaryadm password>
$ /usr/sadm/bin/smrole add -H myHost -- -c "FTP and Backup Operator" \
-n operadm2 -a janedoe -d /export/home/operadm \
-F "Backup/FTP Operator" -p "Media Backup" -p "FTP Management"
Authenticating as user: primaryadm

Type /? for help, pressing <enter> accepts the default denoted by [ ]
Please enter a string value for: password ::      <Type primaryadm password>

Loading Tool: com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost
Login to myHost as user primaryadm was successful.
Download of com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost was successful.

Type /? for help, pressing <enter> accepts the default denoted by [ ]
Please enter a string value for: password ::     <Type operadm2 password>

$ svcadm restart system/name-service-cache
```

`list`サブコマンドを持つ`smrole`コマンドは、新しい役割を表示するのに使用されます。

```
$ /usr/sadm/bin/smrole list --
Authenticating as user: primaryadm

Type /? for help, pressing <enter> accepts the default denoted by [ ]
Please enter a string value for: password ::      <Type primaryadm password>

Loading Tool: com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost
Login to myHost as user primaryadm was successful.
Download of com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost was successful.
root                    0            Superuser
primaryadm            100            Most powerful role
sysadmin              101            Performs non-security admin tasks
operadm               102            Backup Operator
operadm2              103            Backup/FTP Operator
```

Media BackupまたはMedia Restoreプロファイルを含む権利プロファイルによって、ルートファイルシステム全体へのアクセス権を持つ役割が提供されることに注意してください。したがって、管理者はこの権利プロファイルを信頼されるユーザーに割り当てる必要があります。あるいは、管理者はこれらの権利プロファイルを割り当てないことも選択できます。このシナリオでは、スーパーユーザーのみがバックアップと復元を実行できます。

▼ 役割をローカルユーザーに割り当てる方法

この手順では、ローカル役割をローカルユーザーに割り当て、ネームキャッシュデーモンを再起動したあと、ユーザーが役割を引き受ける方法を示します。

分散ネームサービスで役割をユーザーに割り当てる方法については、217 ページの「コマンド行から役割を作成する方法」および232 ページの「役割のプロパティーを変更する方法」を参照してください。

始める前に　ローカル役割は追加済みです (217 ページの「コマンド行から役割を作成する方法」を参照)。Primary Administrator 権利プロファイルを含む役割を引き受けるか、root ユーザーに切り替える必要があります。

1 役割をローカルユーザーに割り当てます。

roleadd コマンドによってローカル役割を追加した場合、この手順が必要です。smrole コマンドと Solaris 管理コンソールを使用して役割を作成したときは、この手順は省略可能です。

```
# usermod -u UID -R rolename login-name
```

`-u` *UID*　ユーザーの UID です。

`-R` *rolename*　ユーザーに割り当てられる役割です。

login-name　ユーザーのログイン名です。

2 変更を有効にするには、ネームサービスキャッシュデーモンを再起動します。

```
# svcadm restart system/name-service-cache
```

Solaris 管理コンソールのインタフェースで役割を追加した場合は、225 ページの「役割の使用 (作業マップ)」に進んでください。そのほかの場合は、次の手順に進みます。

3 (省略可能) 役割アカウントのロックを解除するには、ユーザーがパスワードを作成する必要があります。

roleadd コマンドによってローカル役割を追加した場合、この手順が必要です。

```
% su - rolename
Password:       <Type rolename password>
Confirm Password:       <Retype rolename password>
$
```

例 9–7 コマンド行からのローカル役割の作成と割り当て

この例では、Oracle Solaris の暗号化フレームワークを管理するための役割が作成されます。Crypto Management 権利プロファイルには、ローカルシステムでハードウェアおよびソフトウェアの暗号化サービスを管理する cryptoadm コマンドが含まれます。

```
# roleadd -c "Cryptographic Services manager" \
-g 14 -m /export/home/cryptoadm -u 104 -s pfksh \
-P "Crypto Management" cryptomgt
# usermod -u 1111 -R cryptomgt
# svcadm restart system/name-service-cache
% su - cryptomgt
Password:       <Type cryptomgt password>
Confirm Password:       <Retype cryptomgt password>
$ /usr/ucb/whoami
cryptomgt
$
```

Oracle Solaris の暗号化フレームワークの詳細については、第 13 章「Oracle Solaris の暗号化フレームワーク (概要)」を参照してください。フレームワークの管理方法については、305 ページの「暗号化フレームワークの管理 (作業マップ)」を参照してください。

▼ 役割を監査する方法

役割が行う動作を監査することができます。監査記録に含まれるのは、役割を引き受けたユーザーのログイン名、役割名、および役割が行った動作です。6180:AUE_prof_cmd:profile command:ua,as 監査イベントによって、情報が収集されます。as クラスまたは ua クラスを事前に選択することにより、役割の動作を監査することができます。

1. 監査を計画し、構成ファイルを編集します。

   詳細については、623 ページの「Oracle Solaris 監査 (作業マップ)」を参照してください。

2. **audit_control** ファイルの **flags** 行に **ua** クラスまたは **as** クラスを含めます。

   ```
   ## audit_control file
   flags:lo,as
   naflags:lo
   plugin:name=audit_binfile.so; p_dir=/var/audit
   ```

   ua クラスおよび as クラスには、ほかの監査イベントが含まれます。クラスに含まれる監査イベントについては、audit_event ファイルを参照してください。bsmrecord コマンドを使用することもできます (例 30–27 参照)。

3 監査サービスの構成が終了したら、監査を有効にします。

詳細については、635 ページの「監査サービスの構成と有効化 (手順)」を参照してください。

▼ root ユーザーを役割にする方法

この手順では、ログインユーザーから役割へ root を変更する方法を示します。この手順を完了すると、シングルユーザーモード以外では、root としてシステムに直接ログインできなくなります。root 役割が割り当てられていることと、su コマンドによって root になることが必要です。

root ユーザーを役割に変更することにより、匿名の root ログインを防ぎます。ユーザーは、ログイン後、 root 役割を引き受ける必要があるため、ユーザーのログイン ID が監査サービスに提供され、sulog ファイルに含められます。

この手順では、ローカルユーザーを作成し、このユーザーに root 役割を割り当てます。ユーザーがこの役割を引き受けることを防ぐには、例 9–8 を参照してください。

始める前に root として直接ログインすると、この手順は実行できません。自分のユーザー名でログイン後、su コマンドによって root になる必要があります。

1 通常のユーザーとして、対象のシステムにログインします。

2 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

3 **root** 役割を引き受けることができるローカルユーザーを作成します。

安全のために、少なくとも 1 人のローカルユーザーを root 役割に割り当ててください。

```
$ useradd -c comment -u uid -d homedir username
```

-c *comment* ユーザーについて説明するコメントです。

-d *homedir* ユーザーのホームディレクトリです。このディレクトリはローカルシステム上にあります。

-u *uid* ユーザーの識別番号です。

username　　新しいローカルユーザーの名前です。

```
# useradd -c "JDoe's local account" -u 123 -d /export/home1 jdoe-local
```

4 ユーザーにパスワードを与えます。

```
# passwd -r files jdoe-local
New Password:         <Type password>
Re-enter new Password:      <Retype password>
passwd: password successfully changed for jdoe-local
#
```

5 **root**としてログインしていないことを確認します。

```
# who
jdoe    console      May 24 13:51    (:0)
jdoe    pts/5        May 24 13:51    (:0.0)
jdoe    pts/4        May 24 13:51    (:0.0)
jdoe    pts/10       May 24 13:51    (:0.0)
```

6 **root**ユーザーを役割に変更します。

```
# usermod -K type=role root
```

7 **root**が役割であることを確認します。

user_attrファイルのrootエントリは、次のようになります。

```
# grep root /etc/user_attr
root::::type=role;auths=solaris.*,solaris.grant;profiles=...
```

8 **root**役割を自分のローカルアカウントに割り当てます。

```
# usermod -R root jdoe-local
```

注意 – root役割をユーザーに割り当てないと、シングルユーザーモード以外では誰もスーパーユーザーになれなくなります。シングルユーザーモードに入るには、rootパスワードを入力する必要があります。

9 失敗した場合に返されるようにネームサービスを構成します。

a. 新しい端末ウィンドウを開き、**root**役割を引き受けます。

```
% whoami
jdoe
% su - jdoe-local
Enter password:         <Type jdoe-local password>
% roles
root
% su - root
Enter password:         <Type root password>
#
```

b. **nsswitch.conf** ファイルを編集します。

たとえば、nsswitch.conf ファイルの次のエントリによって、ネームサービスを返すことができます。

```
passwd:  files nis [TRYAGAIN=0 UNAVAIL=return NOTFOUND=return]
group:  files nis [TRYAGAIN=0 UNAVAIL=return NOTFOUND=return]
```

10 (省略可能) **root** 役割をネームサービスで選択されたユーザーアカウントに割り当てます。

この手順については、237 ページの「ユーザーの RBAC プロパティーを変更する方法」を参照してください。

例 9–8 root 役割がシステムの構成に使用されることを防ぐ

この例では、システムの構成にはいくつかの個別の役割を使用するように、サイトのセキュリティーポリシーで要求します。これらの個別の役割はすでに作成され、テストされています。root アカウントがシステムの構成に使用されることを防ぐために、セキュリティー管理者は root を役割に変更し、ただしその役割を割り当てないようにします。シングルユーザーモードでシステムに入ることができるように、root 役割のパスワードは維持します。

管理者はまず、root が役割として割り当て済みでないことを確認します。

```
% whoami
jdoe-local
% su - root
Password: a!2@3#4$5%6^7
# grep roles /etc/user_attr
jdoe-local::::type=normal;roles=secadmin
kdoe-local::::type=normal;roles=sysadmin
```

引き続き root アカウントで、root を役割に変更します。

```
# usermod -K type=role root
```

次に、user_attr ファイル内の root エントリの変更内容を確認します。

```
# grep root /etc/user_attr
root::::type=role;auths=solaris.*,solaris.grant;profiles=...
```

例 9–9 root 役割を変更して root ユーザーに戻す

この例では、管理者はシステムの使用を停止し、スーパーユーザーとしてデスクトップにログインします。システムはすでにネットワークから削除されています。

管理者はまず、root 役割を引き受け、root 役割の割り当てをすべて削除します。

```
% whoami
jdoe-local
% su - root
Password: a!2@3#4$5%6^7
# grep roles /etc/user_attr
jdoe-local::::type=normal;roles=root
kdoe-local::::type=normal;roles=root
# usermod -R "" jdoe-local
# usermod -R "" kdoe-local
# grep roles /etc/user_attr
#
```

引き続き root 役割で、root をユーザーに変更します。

```
# rolemod -K type=normal root
```

次に、user_attr ファイル内の root エントリの変更内容を確認します。

```
# grep root /etc/user_attr
root::::type=normal;auths=solaris.*,solaris.grant;profiles=...
```

注意事項 デスクトップ環境では、root が役割の場合、root として直接ログインすることはできません。このシステム上で root が役割になっていることを示す診断メッセージが表示されます。root 役割を引き受けることのできるローカルアカウントがない場合は、ローカルアカウントを作成します。root としてシングルユーザーモードでシステムにログインし、ローカルユーザーアカウントを作成し、この新しいアカウントに root 役割を割り当てます。次に、この新しいユーザーでログインし、root 役割を引き受けます。

root ユーザーを役割に変更し、次の割り当てのいずれかを実行できない場合、スーパーユーザーになることはできません。

- 有効なユーザーに root 役割を割り当てます。
- root の権利プロファイルと同等の権利プロファイルを、有効なユーザーに割り当てます。Primary Administrator プロファイルは、root の機能に相当する権利プロファイルです。
- root の機能を持った役割を作成し、この役割を有効なユーザーに割り当てます。Primary Administrator プロファイルが割り当てられた役割は、root 役割と同等です。

役割の使用 (作業マップ)

次の作業マップは、役割が割り当てられたあとにその役割を使用する手順を示しています。

作業	説明	参照先
Solaris 管理コンソールを使用します	自分を役割として認証して Solaris 管理コンソールで管理タスクを実行します。	228 ページの「Solaris 管理コンソールで役割を引き受ける方法」
端末ウィンドウで役割を引き受けます	プロファイルシェルでコマンド行の管理タスクを実行します。	226 ページの「端末ウィンドウで役割を引き受ける方法」

役割の使用

役割は、Oracle Solaris のデフォルトの権利プロファイルで設定し、ユーザーに割り当てると、使用できるようになります。役割は、コマンド行で引き受けることができます。Solaris 管理コンソールで、役割は、ローカルおよびネットワーク越しにシステムを管理するために使用することもできます。

▼ 端末ウィンドウで役割を引き受ける方法

始める前に　役割がすでに割り当てられている必要があります。そしてその情報でネームサービスを更新する必要があります。

1. 端末ウィンドウで、引き受けることのできる役割を判断します。

   ```
   % roles
   Comma-separated list of role names is displayed
   ```

2. **su** コマンドを使用して役割を引き受けます。

   ```
   % su - rolename
   Password:       <Type rolename password>
   $
   ```

   su - *rolename* コマンドを実行すると、シェルがその役割のプロファイルシェルに変わります。プロファイルシェルは、セキュリティー属性 (承認、特権、および set ID ビット) を認識します。

3. 役割になっていることを確認します。

   ```
   $ /usr/ucb/whoami
   rolename
   ```

   端末ウィンドウで役割のタスクを実行できるようになりました。

4. **(省略可能) 自分の役割の機能を表示します。**

   この手順については、273 ページの「役割が実行可能な特権付きコマンドを判断する方法」を参照してください。

例 9–10 Primary Administrator 役割を引き受ける

次の例で、ユーザーは Primary Administrator の役割を引き受けます。デフォルトの構成では、この役割はスーパーユーザーと同等です。その後、役割は、その役割のプロファイルシェルに入力されるどのコマンドでも利用可能な特権を確認します。

```
% roles
sysadmin,oper,primaryadm
% su - primaryadm
Password:       <Type primaryadm password>
$ /usr/ucb/whoami       Prompt has changed to role prompt
primaryadm
$ ppriv $$
1200:   pfksh
flags = <none>
        E (Effective): all
        I (Inheritable): basic
        P (Permitted): all
        L (Limit): all
```

特権の詳細については、199 ページの「特権 (概要)」を参照してください。

例 9–11 root 役割を引き受ける

次の例で、ユーザーは root 役割を引き受けます。root 役割は 222 ページの「root ユーザーを役割にする方法」で作成されたものです。

```
% roles
root
% su - root
Password:       <Type root password>
# /usr/ucb/whoami       Prompt has changed to role prompt
root
$ ppriv $$
1200:   pfksh
flags = <none>
        E: all
        I: basic
        P: all
        L: all
```

特権の詳細については、199 ページの「特権 (概要)」を参照してください。

例 9–12 System Administrator 役割を引き受ける

次の例で、ユーザーは System Administrator の役割を引き受けます。Primary Administrator 役割とは対照的に、System Administrator は、効果的に組み合わされた特権の基本セットを備えています。

```
% roles
sysadmin,oper,primaryadm
% su - sysadmin
Password:       <Type sysadmin password>
$ /usr/ucb/whoami       Prompt has changed to role prompt
sysadmin
$ ppriv $$
1200:   pfksh
flags = <none>
        E: basic
        I: basic
        P: basic
        L: all
```

特権の詳細については、199 ページの「特権 (概要)」を参照してください。役割の機能の簡単な説明については、245 ページの「System Administrator 権利プロファイル」を参照してください。

▼ Solaris 管理コンソールで役割を引き受ける方法

Solaris 管理コンソールの GUI で情報を変更するには、管理機能が必要です。役割によって、管理機能が与えられます。情報を表示する場合は、`solaris.admin.usermgr.read` 承認が必要です。Basic Solaris User 権利プロファイルには、この承認が含まれます。

始める前に　ユーザーのプロパティーを変更できる管理役割がすでに割り当てられている必要があります。たとえば、Primary Administrator 役割は、ユーザーまたは役割のプロパティーを変更できます。

1. **Solaris 管理コンソールを起動します。**

   ```
   % /usr/sbin/smc &
   ```

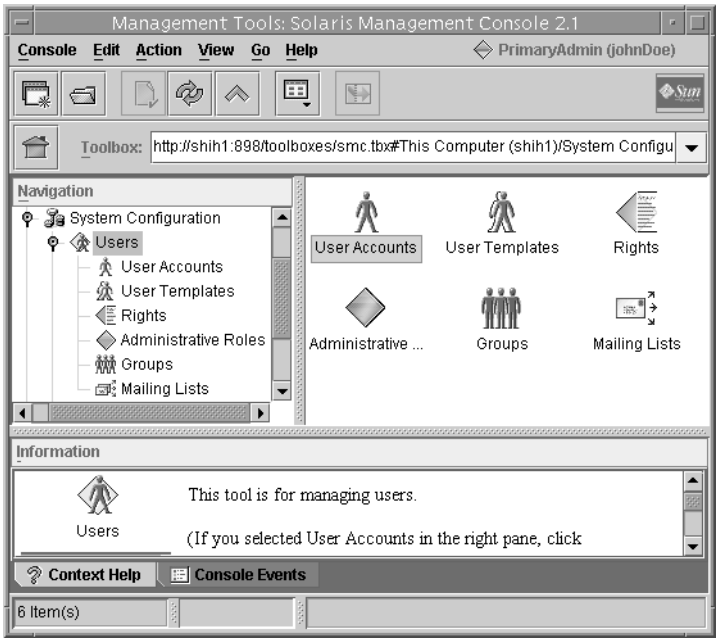
   詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. **操作に必要なツールボックスを選択します。**

   適切なネームサービスの適用範囲のツールまたはコレクションを含むツールボックスに移動し、アイコンをクリックします。適用範囲には、ファイル (ローカル)、NIS、NIS+、および LDAP があります。適切なツールボックスがナビゲーション区画に表示されていない場合は、「コンソール (Console)」メニューから「ツールボックスを開く (Open Toolbox)」を選択して、関連するツールボックスを読み込みます。

3 使用するツールを選択します。

ツールまたはコレクションに移動して、アイコンをクリックします。RBAC 要素を管理するツールは、「ユーザー」ツールにあります (次の図を参照)。



4 「ログイン: ユーザー名 (Login: User Name)」ダイアログボックスで、ユーザー名とパスワードを 入力します。

5 「ログイン: 役割名 (Login: Role)」ダイアログボックスでユーザーを認証させます。

ダイアログボックスの「役割 (Role)」オプションメニューに、割り当てられる役割が表示されます。役割を選択して、役割のパスワードを入力します。

RBAC の管理(作業マップ)

次の作業マップは、役割によるアクセス制御 (RBAC) の初期実装後、RBAC をカスタマイズする手順を示しています。

作業	説明	参照先
役割のパスワードを変更します	承認されたユーザーまたは役割で、別の役割のパスワードを変更します。	230 ページの「役割のパスワードを変更する方法」

作業	説明	参照先
役割のプロパティーを変更します	役割の機能 (特権、特権付きコマンド、プロファイル、承認) を変更します。	232 ページの「役割のプロパティーを変更する方法」
権利プロファイルを作成または変更します	権利プロファイルを作成します。あるいは、権利プロファイルの承認、特権付きコマンド、または補助権利プロファイルを変更します。	234 ページの「権利プロファイルを作成または変更する方法」
ユーザーの管理機能を変更します	役割、権利プロファイル、承認、または特権を通常のユーザーに追加します。	237 ページの「ユーザーの RBAC プロパティーを変更する方法」
レガシーアプリケーションをセキュリティー保護します	レガシーアプリケーションの set ID アクセス権を有効にします。スクリプトでは set ID を有効にしたコマンドを使用できます。必要に応じて、レガシーアプリケーション内で承認を確認できます。	239 ページの「RBAC プロパティーをレガシーアプリケーションに追加する方法」

これらの手順によって、RBAC で使用される要素を管理します。ユーザーの管理手順については、『Solaris のシステム管理 (基本編)』の第 5 章「ユーザーアカウントとグループの管理 (手順)」を参照してください。

RBAC の管理

Solaris 管理コンソールの GUI は、RBAC を管理するための望ましい方法です。

注 – コマンド行インタフェースとグラフィカルユーザーインタフェースを同時に使って、RBAC を管理しないでください。構成に対して矛盾した変更が加えられ、予測していない動作が生じることがあります。両方のツールとも RBAC を管理することができますが、両方を同時に使用することはできません。

▼ 役割のパスワードを変更する方法

始める前に　User Security プロファイルを含んだ役割を引き受けているか、スーパーユーザーに切り替えている必要があります。ある役割のパスワードを変更するには、別の役割を使用する必要があります。使用している役割自体のパスワードを変更することはできません。

- 次のいずれかの方法で、役割のパスワードを変更します。
  - スーパーユーザーとして、または **User Security** 権利プロファイルを含んだ役割で、**passwd** コマンドを実行します。

    ```
    $ passwd  -r naming-service  target-rolename
    ```

`-r` *naming-service* パスワードの変更を `files`、 `nis`、 `nisplus`、 または `ldap` のいずれかのリポジトリに適用します。リポジトリを指定しない場合は、`files` のパスワードが変更されます。

target-rolename 変更する既存の役割の名前です。

コマンドオプションの詳細については、`passwd(1)` のマニュアルページを参照してください。

- **Solaris 管理コンソールでパスワードを変更します。**

  コンソールの起動については、228 ページの「Solaris 管理コンソールで役割を引き受ける方法」を参照してください。

  a. スーパーユーザーとして、または **User Security** 権利プロファイルを含んだ役割で、コンソールにログインします。

     対象の役割ではログインできません。

  b. 適切な適用範囲を選択します。

     適用範囲として `Files` を選択すると、ローカルシステム上の役割のパスワードが変更されます。適用範囲として `LDAP` を選択すると、LDAP ネームサービス内の役割のパスワードが変更されます。

  c. 「管理役割 **(Administrative Roles)**」に移動し、左側の区画の指示に従って操作します。

     詳細は、オンラインヘルプを参照してください。

- スーパーユーザーとして、または **User Security** 権利プロファイルを含んだ役割で、**`smrole`** コマンドを **`modify`** サブコマンドとともに実行します。

  このコマンドは、Solaris 管理コンソールサーバーのクライアントとして動作します。

  ```
  $ /usr/sadm/bin/smrole -D domain-name -r admin-role -l <Type admin-role password> \
  modify -- -n target-rolename -P password
  ```

  `-D` *domain-name* 管理対象のドメインの名前です。

  `-r` *admin-role* 対象の役割を変更できる管理役割の名前です。管理役割は `solaris.admin.usermgr.pswd` 承認を得る必要があります。管理役割と対象の役割は同じではいけません。

  `-l` *admin-role* のパスワードに対するプロンプトです。

  `--` 認証オプションとサブコマンドオプションの間に必要な区切り文字です。

  `-n` *target-rolename* 対象の役割の名前です。

-P *password* *target-rolename* の新しいパスワードです。

コマンドオプションの一覧については、smrole(1M) のマニュアルページを参照してください。

例 9–13 passwd コマンドによるローカル役割のパスワードの変更

この例では、スーパーユーザーがローカルの operadm 役割のパスワードを変更します。

```
# passwd -r files  operadm
New password:       Type new password
Re-enter new password:       Retype new password
```

例 9–14 LDAP リポジトリ内の役割のパスワードを変更する

この例では、Primary Administrator 役割が LDAP ディレクトリサービス内の operadm 役割のパスワードを変更します。

```
$ passwd -r ldap operadm
New password:       Type new password
Re-enter new password:       Retype new password
```

例 9–15 smrole modify コマンドによる役割のパスワードの変更

この例では、管理者が Solaris 管理コンソールサーバーに接続し、NIS ドメイン内の operadm のパスワードを変更します。管理者がパスワードを指定せずに Return キーを押すと、「New Password:」プロンプトが表示されます。

```
$ /usr/sadm/bin/smrole -D nis:/examplehost/example.domain \
-r primaryadm -l <Type primaryadm password> \
modify -- -n operadm -P       Press the Return key
New Password: a!2@3#4$5%6*7
$
```

▼ 役割のプロパティーを変更する方法

始める前に Primary Administrator 権利プロファイルを含む役割を引き受けるか、root ユーザーに切り替えて役割のプロパティーを変更する必要があります。役割のプロパティーには、パスワード、権利プロファイル、および承認が含まれます。

注 – 役割のパスワードプロパティーを変更する方法については、230 ページの「役割のパスワードを変更する方法」を参照してください。

- 次のいずれかの方法で、役割のプロパティーを変更します。

  - **Solaris管理コンソールの「ユーザー」ツールを使用します。**

    コンソールの起動については、228ページの「Solaris管理コンソールで役割を引き受ける方法」を参照してください。左側の区画の指示に従って、管理役割の役割を変更します。詳細は、オンラインヘルプを参照してください。

  - **`rolemod`コマンドを使用します。**

    このコマンドは、ローカルのネームサービスで定義される役割の属性を変更します。

    ```
    $ rolemod -c comment -P profile-list rolename
    ```

    `-c` *comment* 役割の機能を説明する新しいコメントです。

    `-P` *profile-list* 役割に含まれるプロファイルのリストです。このリストでプロファイルの現在のリストが置き換えられます。

    rolename 変更する既存のローカル役割の名前です。

    コマンドオプションの詳細については、`rolemod(1M)`のマニュアルページを参照してください。

  - **`modify`サブコマンドを持つ`smrole`コマンドを使用します。**

    このコマンドは、NIS、NIS+、LDAPなどの分散ネームサービスで役割の属性を変更します。Solaris管理コンソールサーバーのクライアントとして動作します。

    ```
    $ /usr/sadm/bin/smrole -D domain-name \
    -r admin-role -l <Type admin-role password> \
    modify -- -n rolename -r username -u username
    ```

    `-D` *domain-name* 管理対象のドメインの名前です。

    `-r` *admin-role* 役割を変更できる管理役割の名前です。管理役割は`solaris.role.assign`承認を得る必要があります。引き受けた役割を変更する場合、役割は`solaris.role.delegate`承認を得る必要があります。

    `-l` *admin-role*のパスワードに対するプロンプトです。

    `--` 認証オプションとサブコマンドオプションの間に必要な区切り文字です。

    `-n` *rolename* 新しい役割の名前です。

    `-r` *username* *rolename*を引き受けることができなくなったユーザーの名前です。

    `-u` *username* *rolename*を引き受けることができるようになったユーザーの名前です。

コマンドオプションの詳細については、smrole(1M) のマニュアルページを参照してください。

例 9–16 rolemod コマンドによるローカル役割のプロパティーの変更

この例では、FTP Management 権利プロファイルを含むように operadm 役割を変更します。

```
$ rolemod -c "Handles printers, backup, and FTP" \
-P "Operator,FTP Management,All" operadm
```

これらの権利プロファイルは、policy.conf ファイルを介して与えられたプロファイルに追加されます。

例 9–17 smrole modify コマンドによるローカル役割のプロパティーの変更

次の例では、FTP Management 権利プロファイルを追加するように operadm 役割を変更します。

```
$ /usr/sadm/bin/smrole -r primaryadm -l <Type primaryadm password> \
modify -- -n operadm -c "Handles printers, backup, and FTP" \
-p "FTP Management"
```

例 9–18 smrole modify コマンドによるドメインの役割の変更

次の例では、clockmgr 役割を変更します。ID が 108 の NIS ユーザーは役割を引き受けることができなくなります。ID が 110 の NIS ユーザーは、clockmgr の役割を引き受けることができます。

```
$ /usr/sadm/bin/smrole -D nis:/examplehost/example.domain \
-r primaryadm -l <Type primaryadm password> \
modify -- -n clockmgr -r 108 -u 110
```

▼ 権利プロファイルを作成または変更する方法

権利プロファイルは、役割のプロパティーです。prof_attr データベースに必要な権利プロファイルが含まれていないときには、権利プロファイルを作成または変更してください。権利プロファイルの詳細については、196 ページの「RBAC の権利プロファイル」を参照してください。

始める前に　権利プロファイルを作成または変更するには、Primary Administrator の役割を引き受けているか、スーパーユーザーに切り替えている必要があります。

- 次のいずれかの方法で、権利プロファイルを作成または変更します。

  - **Solaris管理コンソールの「ユーザー」ツールを使用します。**

    コンソールの起動については、228ページの「Solaris管理コンソールで役割を引き受ける方法」を参照してください。左側の区画の指示に従って、「権利」の権利プロファイルを作成または変更します。詳細は、オンラインヘルプを参照してください。

  - **`smprofile`コマンドを使用します。**

    このコマンドによって、権利プロファイルの追加、変更、一覧表示、または削除を行うことができます。このコマンドは、ファイル、およびNIS、NIS+、LDAPなどの分散ネームサービスで機能します。`smprofile`コマンドは、Solaris管理コンソールサーバーのクライアントとして動作します。

    ```
    $ /usr/sadm/bin/smprofile -D domain-name \
    -r admin-role -l <Type admin-role password> \
    add | modify -- -n profile-name \
    -d description  -m help-file -p supplementary-profile
    ```

`-D` *domain-name*	管理対象のドメインの名前です。
`-r` *admin-role*	役割を変更できる管理役割の名前です。管理役割は`solaris.role.assign`承認を得る必要があります。引き受けた役割を変更する場合、役割は`solaris.role.delegate`承認を得る必要があります。
`-l`	*admin-role*のパスワードに対するプロンプトです。
`--`	認証オプションとサブコマンドオプションの間に必要な区切り文字です。
`-n` *profile-name*	新しいプロファイルの名前です。
`-d` *description*	プロファイルの簡単な説明です。
`-m` *help-file*	作成した`/usr/lib/help/profiles/locale/C`ディレクトリに配置したHTMLヘルプファイルの名前です。
`-p` *supplementary-profile*	この権利プロファイルに含まれる既存の権利プロファイルの名前です。複数の`-p` *supplementary-profile*のオプションを指定することができます。

    コマンドオプションの詳細については、`smprofile(1M)`のマニュアルページを参照してください。

例 9–19 コマンド行から権利プロファイルを変更する

次の例では、Network Management 権利プロファイルを Network Security 権利プロファイルの補助プロファイルにします。Network Security プロファイルを含む役割は、ネットワークおよびホストを構成できるようになり、加えてセキュリティー関連のコマンドを実行します。

```
$ /usr/sadm/bin/smprofile -D nisplus:/example.host/example.domain \
-r primaryadm -l <Type primaryadm password> \
modify -- -n "Network Security" \
-d "Manage network and host configuration and security" \
-m RtNetConfSec.html -p "Network Management"
```

管理者は、新しいヘルプファイル、`RtNetConfSec.html` を作成し、それを`/usr/lib/help/profiles/locale/C` ディレクトリに配置してから、このコマンドを実行します。

例 9–20 権利ツールを使用して新しい権利プロファイルを作成する

次の表では、「Build Administrator」と呼ばれる仮想権利プロファイルのサンプルデータを示します。この権利プロファイルには、サブディレクトリ `/usr/local/swctrl/bin` のコマンドが含まれます。これらのコマンドの実効 UID は `0` です。Build Administrator 権利プロファイルは、ソフトウェア開発のビルドとバージョンを管理する管理者が使用します。

タブ	フィールド	例
基本 (General)	名前	Build Administrator
	説明	ソフトウェアのビルドとバージョンの管理用。
	ヘルプファイル名 (Help File Name)	`BuildAdmin.html`
コマンド (Commands)	ディレクトリを追加 (Add Directory)	「ディレクトリを追加 (Add Directory)」をクリックし、ダイアログボックスに `/usr/local/swctrl/bin` と入力して、「了解 (OK)」をクリックします。
	拒否されたコマンド (Commands Denied) / 許可されたコマンド (Commands Permitted)	`/usr/local/swctrl/bin` を「許可されたコマンド (Commands Permitted)」列に移動します。
	セキュリティー属性を設定 (Set Security Attributes)	`/usr/local/swctrl/bin` を選択し、「セキュリティー属性を 設定 (Set Security Attributes)」をクリックして、 Effective UID = `root` を設定します。

タブ	フィールド	例
承認	含まれない承認 (Authorizations Excluded) / 含まれる承認 (Authorizations Included)	承認なし
補助権利 (Supplementary Rights)	含まれない権利 (Rights Excluded) / 含まれる権利 (Rights Included)	補助権利プロファイルなし

注意事項 権利プロファイルによって期待する機能を持つ役割が提供されない場合は、次を確認します。

- 役割の権利プロファイルが、GUIで最も権限のあるものから最も権限のないものへとリストされていること。

  たとえば、`All`権利プロファイルがリストの最上部にある場合、セキュリティー属性で実行されるコマンドはありません。セキュリティー属性を指定したコマンドを含むプロファイルは、リストで`All`権利プロファイルの前に来なければなりません。
- コマンドが、役割の権利プロファイルで複数回リストされていること。その場合、コマンドの最初のインスタンスに必要なセキュリティー属性がすべて指定されていること。

  たとえば、コマンドはそのコマンドの特定のオプションのために特権を必要とする可能性があります。特権を必要とするオプションが成功するには、リストの一番上にある権利プロファイルのコマンドの最初のインスタンスに特権を割り当てる必要があります。
- 役割の権利プロファイルのコマンドに適切なセキュリティー属性を指定していること。

  たとえば、ポリシーが`suser`のとき、`euid=0`ではなく、`uid=0`である必要があるコマンドもあります。
- ネームサービスのキャッシュ、`svc:/system/name-service-cache`を再起動していること。

  `nscd`デーモンには長い有効期間を設定することができます。デーモンを再起動して、現在のデータでネームサービスを更新します。

▼ ユーザーのRBACプロパティーを変更する方法

ユーザーのプロパティーには、パスワード、権利プロファイル、役割、および承認が含まれます。ユーザーに管理機能を与える最も安全な方法として、ユーザーに役割を与えます。詳細については、198ページの「セキュリティー属性を直接割り当てる場合に考慮すべきセキュリティー事項」を参照してください。

始める前に　Primary Administrator 権利プロファイルを含む役割を引き受けるか、root ユーザーに切り替える必要があります。

- 次のいずれかの方法で、ユーザーの **RBAC** プロパティーを変更します。
  - **Solaris** 管理コンソールの「ユーザー」ツールを使用します。

    コンソールの起動については、228 ページの「Solaris 管理コンソールで役割を引き受ける方法」を参照してください。左側の区画の指示に従って、「ユーザーアカウント (User Accounts)」のユーザーを変更します。詳細は、オンラインヘルプを参照してください。

    ヒント – 承認、特権、または権利プロファイルをユーザーに直接割り当てることはお勧めできません。役割をユーザーに割り当てることが望ましい方法です。その後、ユーザーが、特権付きの操作を実行するための役割を引き受けます。

  - **usermod** コマンドを使用します。

    このコマンドは、ローカルのネームサービスで定義されるユーザーの属性を変更します。

    ```
    $ usermod -R rolename username
    ```

    -R *rolename*　既存のローカル役割の名前です。

    username　変更する既存のローカルユーザーの名前です。

    コマンドオプションの詳細については、usermod(1M) のマニュアルページを参照してください。

  - **modify** サブコマンドを持つ **smuser** コマンドを使用します。

    このコマンドは、NIS、 NIS+、 LDAP などの分散ネームサービスでユーザーの属性を変更します。Solaris 管理コンソールサーバーのクライアントとして動作します。

    ```
    $ /usr/sadm/bin/smuser -D domain-name \
    -r admin-role -l <Type admin-role password> \
    modify -- -n username -a rolename
    ```

    -D *domain-name*　管理対象のドメインの名前です。

    -r *admin-role*　役割を変更できる管理役割の名前です。管理役割は solaris.role.assign 承認を得る必要があります。引き受けた役割を変更する場合、役割は solaris.role.delegate 承認を得る必要があります。

    -l　*admin-role* のパスワードに対するプロンプトです。

    --　認証オプションとサブコマンドオプションの間に必要な区切り文字です。

-n *username*　　*rolename* を割り当てられるユーザーの名前です。

-a *rolename*　　*username* に割り当てる役割の名前です。複数の -a *rolename* オプションを指定することができます。

コマンドオプションの詳細については、smuser(1M) のマニュアルページを参照してください。

例 9–21 コマンド行からのローカルユーザーの RBAC プロパティーの変更

この例では、ユーザー jdoe が System Administrator の役割を引き受けることができるようになります。

```
$ usermod -R sysadmin jdoe
```

この役割は、ユーザーが引き受けることのできる役割に追加されます。

例 9–22 smuser コマンドを使用したユーザーの RBAC プロパティーの変更

この例では、ユーザー jdoe に、System Administrator と Operator の 2 つの役割が割り当てられます。ユーザーと役割はローカルに定義されるため、-D オプションは必要ありません。

```
$ /usr/sadm/bin/smuser -r primaryadm -l <Type primaryadm password> \
modify -- -n jdoe -a sysadmin -a operadm
```

次の例では、ユーザーは NIS ネームサービスで定義されます。したがって、-D オプションは必要です。2 つの役割が、ネームサービスで定義されます。一方の役割、root は、ローカルに定義されます。

```
$ /usr/sadm/bin/smuser -D nis:/examplehost/example.domain \
-r primaryadm -l <Type primaryadm password> \
modify -- -n jdoe -a sysadmin -a operadm -a root
```

▼ RBAC プロパティーをレガシーアプリケーションに追加する方法

レガシーアプリケーションは、コマンドまたはコマンドのセットです。セキュリティー属性は、権利プロファイルのコマンドごとに設定します。その後、権利プロファイルを役割に含めます。役割を引き受けるユーザーは、セキュリティー属性を指定したレガシーアプリケーションを実行することができます。

Solaris 管理コンソールにレガシーアプリケーションを追加する方法については、『Solaris のシステム管理 (基本編)』の「Solaris 管理コンソールにツールを追加する」を参照してください。

始める前に　権利プロファイルのコマンドのセキュリティー属性を変更するには、Primary Administrator の役割を引き受けているか、スーパーユーザーに切り替えている必要があります。

1. **Solaris 管理コンソールの「ユーザー」ツールを使用します。**

   コンソールの起動については、228 ページの「Solaris 管理コンソールで役割を引き受ける方法」を参照してください。左側の区画の指示に従って、「権利」の権利プロファイルを変更します。詳細は、オンラインヘルプを参照してください。

2. **レガシーアプリケーションを実装するコマンドにセキュリティー属性を追加します。**

   レガシーアプリケーションにセキュリティー属性を追加するときは、コマンドに追加する場合と同じ方法で追加します。セキュリティー属性を指定したコマンドを権利プロファイルに追加する必要があります。レガシーコマンドに対して、`euid=0` または `uid=0` のセキュリティー属性を指定します。手順の詳細については、234 ページの「権利プロファイルを作成または変更する方法」を参照してください。

3. **レガシーアプリケーションを権利プロファイルに追加したあと、権利プロファイルを役割のプロファイルリストに含めます。**

   権利プロファイルを役割に追加する方法については、232 ページの「役割のプロパティーを変更する方法」を参照してください。

例 9–23　スクリプトのコマンドへのセキュリティー属性の追加

スクリプトのコマンドが `setuid` ビットまたは `setgid` ビットセットを持つ必要がある場合、実行可能なスクリプトおよびコマンドに対して、権利プロファイルでセキュリティー属性を追加する必要があります。その後、権利プロファイルを役割に含め、含めた役割をユーザーに割り当てます。ユーザーが、役割を引き受け、スクリプトを実行すると、コマンドはそのセキュリティー属性で実行されます。

セキュリティー属性をコマンドまたはシェルスクリプトに追加する方法については、234 ページの「権利プロファイルを作成または変更する方法」を参照してください。

例 9–24　スクリプトまたはプログラム内の承認の確認

承認用のスクリプトを用意するには、`auths` コマンドに基づいたテストを追加する必要があります。このコマンドの詳細については、`auths(1)` のマニュアルページを参照してください。

たとえば、次の行では、`$1` 引数に指定した承認がユーザーに与えられているかどうかをテストします。

```
if [ `/usr/bin/auths|/usr/xpg4/bin/grep $1` ]; then
        echo Auth granted
else
        echo Auth denied
fi
```

万全を期すために、テストにはワイルドカードを使用する別の承認を確認する論理を含めるようにしてください。たとえば、solaris.admin.usermgr.write 承認がユーザーに与えられているかどうかをテストするには、次の文字列を確認します。

- solaris.admin.usermgr.write
- solaris.admin.usermgr.*
- solaris.admin.*
- solaris.*

プログラムを作成している場合は、getauthattr() 関数を使用して、承認をテストします。

◆ ◆ ◆ 第 10 章

10 役割によるアクセス制御 (参照)

この章は、RBAC に関する参照資料です。この章の内容は次のとおりです。

RBAC の使用方法については、第 9 章「役割によるアクセス制御の使用 (手順)」を参照してください。概要については、188 ページの「役割によるアクセス制御 (概要)」を参照してください。

権利プロファイルの内容

この節では、いくつかの標準的な権利プロファイルについて説明します。権利プロファイルには、承認、セキュリティー属性を指定したコマンド、および補助権利プロファイルを含めることができます。権利プロファイルは、最も権限のあるものから最も権限のないものへとリストされます。権利プロファイルをサイトの役割に配布する際の方法については、211 ページの「RBAC の実装を計画する方法」を参照してください。

- 「**Primary Administrator**」権利プロファイル – 1 つのプロファイルでのスーパーユーザーの機能を提供します。
- 「**System Administrator**」権利プロファイル – セキュリティーに関係しないほとんどのタスクを実行できるプロファイルを提供します。このプロファイルには、権限のある役割を作成するためにいくつかのほかのプロファイルが含まれます。
- 「**Operator**」権利プロファイル – ファイルおよびオフ欄メディアを管理するための制限された機能を提供します。 このプロファイルには、単純な役割を作成するための補助権利プロファイルが含まれます。

- 「**Printer Management**」権利プロファイル – 印刷を処理するための限られた数のコマンドと承認を提供します。このプロファイルは、単一の管理領域を対象とする複数のプロファイルのうちの 1 つです。
- 「**Basic Solaris User**」権利プロファイル – セキュリティーポリシーの範囲内でユーザーがシステムを使用できるようにします。このプロファイルは、デフォルトで `policy.conf` ファイル内にリストされます。
- 「**All**」権利プロファイル – 役割に対して、セキュリティー属性を指定していないコマンドへのアクセスを提供します。

それぞれの権利プロファイルには、関連するヘルプファイルが用意されています。ヘルプファイルは、HTML 形式で、カスタマイズが可能です。ヘルプファイルは、`/usr/lib/help/profiles/locale/C` ディレクトリにあります。

Primary Administrator 権利プロファイル

Primary Administrator 権利プロファイルには、システム上で最も強力な役割が割り当てられます。Primary Administrator 権利プロファイルを含む役割は、スーパーユーザーの機能を持ちます。

- `solaris.*` 承認は、実質的に Oracle Solaris ソフトウェアから提供されるすべての承認を割り当てます。
- `solaris.grant` 承認は、任意の権利プロファイル、役割、またはユーザーに任意の承認を割り当てます。
- `*:uid=0;gid=0` のコマンド割り当ては、`UID=0` および `GID=0` ですべてのコマンドを実行します。

ヘルプファイル `RtPriAdmin.html` は、必要に応じて、使用するサイトに合わせてカスタマイズできます。ヘルプファイルは、`/usr/lib/help/profiles/locale/C` ディレクトリに格納されています。

Primary Administrator 権利プロファイルがサイトのセキュリティーポリシーと矛盾する場合は、このプロファイルを変更したり、割り当てないようにしたりすることもできます。ただし、Primary Administrator 権利プロファイルのセキュリティー機能は、ほかの権利プロファイルの処理に必要となります。この場合、それらのほかの権利プロファイルを役割に割り当てます。

表 10–1　Primary Administrator 権利プロファイルの内容

目的	内容
すべての管理タスクを実行する	コマンド: `*:uid=0;gid=0` 承認: `solaris.*`、 `solaris.grant` ヘルプファイル: `RtPriAdmin.html`

System Administrator 権利プロファイル

System Administrator 権利プロファイルは、System Administrator 役割用に設計されています。System Administrator では、Primary Administrator の強力な機能を持たないため、ワイルドカードは使用できません。その代わり、このプロファイルは、セキュリティーを対象外とする、一連の個別の補助的な管理権利プロファイルです。補助権利プロファイルの 1 つからのセキュリティー属性を指定したコマンドを示します。

All 権利プロファイルは、補助権利プロファイルのリストの最後にあります。

表 **10–2** System Administrator 権利プロファイルの内容

目的	内容
セキュリティーに関係しない管理タスクを実行する	補助権利プロファイル: Audit Review、 Printer Management、 Cron Management、 Device Management、 File System Management、 Mail Management、 Maintenance and Repair、 Name Service Management、 Network Management、 Object Access Management、 Process Management、 Software Installation、 Project Management、 User Management、 All ヘルプファイル: `RtSysAdmin.html`
補助プロファイルの 1 つからのコマンド	**Object Access Management** 権利プロファイル、 `solaris` ポリシー: `/usr/bin/chgrp:privs=file_chown`、 `/usr/bin/chmod:privs=file_chown`、 `/usr/bin/chown:privs=file_chown`、 `/usr/bin/setfacl:privs=file_chown` `suser` ポリシー: `/usr/bin/chgrp:euid=0`、 `/usr/bin/chmod:euid=0`、 `/usr/bin/chown:euid=0`、 `/usr/bin/getfacl:euid=0`、 `/usr/bin/setfacl:euid=0`

Operator 権利プロファイル

Operator 権利プロファイルは、権限の弱いプロファイルで、バックアップとプリンタ管理を行います。ファイルの復元は、セキュリティーに影響します。したがって、このプロファイルでは、デフォルトではファイルの復元機能は含まれていません。

表 **10–3** Operator 権利プロファイルの内容

目的	内容
単純な管理タスクを実行する	補助権利プロファイル: Printer Management、 Media Backup、 All ヘルプファイル: `RtOperator.html`

Printer Management 権利プロファイル

Printer Management は標準的な権利プロファイルで、特定のタスク領域用に設計されています。このプロファイルには、承認とコマンドが含まれます。次の表では、使用できるコマンドの一部を示します。

表 10–4 Printer Management 権利プロファイルの内容

目的	内容
プリンタ、デーモン、スプール処理を管理する	承認: `solaris.print.*`、 `solaris.label.print`、`solaris.admin.printer.delete`、`solaris.admin.printer.modify`、`solaris.admin.printer.read`、`solaris.smf.manage.discovery.printers.*`、`solaris.smf.value.discovery.printers.*` コマンド: `/usr/lib/lp/local/lpadmin:uid=lp;gid =lp`、`/usr/sbin/lpfilter:euid=lp;uid=lp`、`/usr/sbin/lpforms:euid=lp`、`/usr/sbin/lpusers:euid=lp`、 `/usr/sbin/ppdmgr:euid=0` ヘルプファイル: `RtPrntMngmnt.html`

Basic Solaris User 権利プロファイル

デフォルトでは、Basic Solaris User 権利プロファイルは、`policy.conf` ファイルによってすべてのユーザーに自動的に割り当てられます。このプロファイルでは、通常の操作に使用する基本的な承認を与えます。Basic Solaris User 権利プロファイルを使用するときは、サイトのセキュリティー要件を考慮する必要があります。高いセキュリティーを必要とするサイトでは、このプロファイルを `policy.conf` ファイルから削除することをお勧めします。

表 10–5 Basic Solaris User 権利プロファイルの内容

目的	内容
すべてのユーザーに自動的に権限を割り当てる	承認: `solaris.profmgr.read`、 `solaris.jobs.user`、 `solaris.mail.mailq`、 `solaris.device.mount.removable`、 `solaris.admin.usermgr.read`、 `solaris.admin.logsvc.read`、 `solaris.admin.fsmgr.read`、 `solaris.admin.serialmgr.read`、 `solaris.admin.diskmgr.read`、 `solaris.admin.procmgr.user`、 `solaris.compsys.read`、 `solaris.admin.printer.read`、 `solaris.admin.prodreg.read`、 `solaris.admin.dcmgr.read`、 `solaris.snmp.read`、 `solaris.project.read`、 `solaris.admin.patchmg.read`、 `solaris.network.hosts.read`、 `solaris.admin.volmgr.read` 補助権利プロファイル: All ヘルプファイル: `RtDefault.html`

All 権利プロファイル

All 権利プロファイルは、すべてのコマンドを使用できるようにワイルドカードを使用したプロファイルです。この権利プロファイルは、ほかのプロファイルに明示的に割り当てられていないすべてのコマンドにアクセスできる役割です。All 権利プロファイルまたはワイルドカードを使用するその他の権利プロファイルを使用しないと、役割は明示的に割り当てられているコマンド以外にはアクセスできません。このような制限された一連のコマンドは、あまり実用的でありません。このプロファイルには承認は含まれません。

All 権利プロファイルを使用する場合は、最後に割り当ててください。それによって、ほかの権利プロファイルの明示的なセキュリティー属性割り当てが確実に適用できます。

表 10–6 All 権利プロファイルの内容

目的	内容
ユーザーまたは役割として任意のコマンドを実行する	コマンド: `*` ヘルプファイル: `RtAll.html`

権利プロファイルの順序

権利プロファイルのコマンドは、発生順に解釈されます。最初に発生したコマンドが、役割またはユーザーに対して使用されるコマンドの唯一のバージョンです。さまざまな権利プロファイルが、同一のコマンドを含むことができます。したがって、プロファイルのリスト内の権利プロファイルの順序が重要になってきます。ほとんどの機能を持つ権利プロファイルが先頭に来るようにします。

権利プロファイルは、Solaris 管理コンソールの GUI および prof_attr ファイルでリストされます。Solaris 管理コンソールの GUI では、ほとんどの機能を持つ権利プロファイルが、割り当てられた権利プロファイルのリストの一番上のプロファイルになります。prof_attr ファイルでは、ほとんどの機能を持つ権利プロファイルが、補助プロファイルのリストの最初に来ます。この配置において、セキュリティー属性を指定したコマンドがセキュリティー属性を指定していない同一のコマンドの前にリストされます。

権利プロファイルの内容の表示

Solaris 管理コンソールの権利ツールを使用して、権利プロファイルの内容を検査することもできます。

prof_attr および exec_attr ファイルでは、より細分化されて表示されます。prof_attr ファイルには、システムで定義されたすべての権利プロファイルの名前が含まれます。このファイルには、プロファイルごとの承認、特権、および補助権利プロファイルも含まれます。exec_attr ファイルには、権利プロファイルの名前と、セキュリティー属性を指定した権利プロファイルのコマンドが含まれます。

承認の命名と委託

RBAC の「承認」は、役割またはユーザーに許可できる個別の権限です。RBAC に準拠したアプリケーションによって承認が確認されてから、ユーザーはアプリケーションまたはアプリケーションの特定の操作へのアクセス権を取得します。この承認の確認は、従来の UNIX アプリケーションが行なっていた UID=0 のテストに代わるものです。

承認の命名規則

承認には、RBAC の内部およびファイル内で使用される名前があります。たとえば solaris.admin.usermgr.pswd などの承認名があります。また、グラフィカルユーザーインタフェース (GUI) に表示される短い説明もあります。たとえば、Change Passwords は、solaris.admin.usermgr.pswd 承認の説明です。

承認名の書式は、インターネット名と逆の順序になり、サプライヤ、被認証者領域、任意の下位領域、および承認の機能で構成されます。承認名の区切り文字はドット (.) です。たとえば、com.xyzcorp.device.access のように指定します。ただし、Sun から許可される承認では、インターネット名の代わりに接頭辞 solaris が使用されます。システム管理者は、承認を階層方式で適用することができます。ワイルドカード (*) は、ドットの右側の任意の文字列を表すことができます。

承認レベルの違いの例

ここでは、承認の使用方法の例を示します。 Operator 役割のユーザーは、多くの場合、solaris.admin.usermgr.read 承認に制限されます。この承認では、ユーザーの構成ファイルに対する読み取り権は許可されますが、書き込み権は許可されません。System Administrator 役割では、solaris.admin.usermgr.read 承認と、ユーザーのファイルを変更できる solaris.admin.usermgr.write 承認が許可されます。ただし、System Administrator には、solaris.admin.usermgr.pswd 承認が許可されないため、パスワードは変更できません。Primary Administrator では、これらの 3 つの承認がすべて許可されます。

Solaris 管理コンソールのユーザーツールのパスワードを変更するには、solaris.admin.usermgr.pswd 承認が必要です。この承認は、 smuser、 smmultiuser、 および smrole コマンドのパスワード変更オプションを使用するときにも必要になります。

承認での委託権限

接尾辞が grant の承認が許可されたユーザーまたは役割は、割り当てられている承認のうち同じ接頭辞を持つ任意の承認を、ほかのユーザーに委託することができます。

たとえば、solaris.admin.usermgr.grant 承認と solaris.admin.usermgr.read 承認を持つ役割は、solaris.admin.usermgr.read 承認をほかのユーザーに委託できます。solaris.admin.usermgr.grant 承認と solaris.admin.usermgr.* 承認を持つ役割は、solaris.admin.usermgr 接頭辞を持つ任意の承認をほかのユーザーに委託できます。

RBACをサポートするデータベース

次の 4 つのデータベースには、RBAC 要素のデータが格納されます。

- 拡張ユーザー属性のデータベース (user_attr) – ユーザーと役割を、承認、特権、および権利プロフィールに関連付けます
- 権利プロファイル属性のデータベース (prof_attr) – 権利プロファイルを定義し、その権利プロファイルに割り当てられた承認とキーワードを指定し、関連するヘルプファイルを指定します
- 承認属性のデータベース (auth_attr) – 承認とその属性を定義し、関連するヘルプファイルを指定します
- 実行属性のデータベース (exec_attr) – 特定の権利プロファイルに割り当てられたセキュリティー属性を持つコマンドを指定します

policy.conf データベースには、すべてのユーザーに適用される承認、特権、および権利プロファイルが含まれます。詳細については、256 ページの「policy.conf ファイル」を参照してください。

RBAC データベースの関係

各 RBAC データベースには、*key= value* という構文を使用して、値を格納します。この方式は、将来のデータベースの拡張に対応します。また、ポリシーが認識できないキーワードが検出された場合にも対応できます。*key=value* の内容によって、ファイルがリンクされます。4 つのデータベースの互いにリンクした次のエントリは、RBAC データベースの相互関係を示しています。

例 10–1　RBAC データベースの接続を示す

次の例で、ユーザー jdoe は、役割 filemgr を割り当てられることにより、File System Management 権利プロファイルの機能を取得します。

1. ユーザー jdoe には、user_attr データベースの jdoe ユーザーエントリの役割 filemgr が割り当てられます。

   ```
   # user_attr - user definition
   jdoe::::type=normal;roles=filemgr
   ```

2. 役割 filemgr には、user_attr データベースの役割のエントリの権利プロファイル File System Management が割り当てられます。

   ```
   # user_attr - role definition
   filemgr::::profiles=File System Management;type=role
   ```

   ユーザーと役割は、ローカルシステムの passwd ファイルおよび shadow ファイル、または分散ネームサービスの同等のデータベースに一意に定義されます。

3. File System Management 権利プロファイルは prof_attr データベースに定義されています。また、このデータベースは File System Management エントリに 3 つの承認のセットを割り当てます。

   ```
   # prof_attr - rights profile definitions and assigned authorizations
   File System Management:::Manage, mount, share file systems:
   help=RtFileSysMngmnt.html;
   auths=solaris.admin.fsmgr.*,solaris.admin.diskmgr.*,solaris.admin.volmgr.*
   ```

4. これらの承認は、auth_attr データベースに定義されています。

   ```
   # auth_attr - authorization definitions
   solaris.admin.fsmgr.:::Mounts and Shares::help=AuthFsmgrHeader.html
   solaris.admin.fsmgr.read:::View Mounts and Shares::help=AuthFsmgrRead.html
   solaris.admin.fsmgr.write:::Mount and Share Files::help=AuthFsmgrWrite.html
   ```

5. File System Management 権利プロファイルには、exec_attr データベースでセキュリティー属性を指定したコマンドが割り当てられます。

   ```
   # exec_attr - rights profile names with secured commands
   File System Management:suser:cmd:::/usr/sbin/mount:uid=0
   File System Management:suser:cmd:::/usr/sbin/dfshares:euid=0
   ```

例 10–1 RBACデータベースの接続を示す (続き)

```
...
File System Management:solaris:cmd:::/usr/sbin/mount:privs=sys_mount
...
```

RBACデータベースおよびネームサービス

RBACデータベースのネームサービスの適用範囲は、ローカルホストにのみ適用することができます。また、適用範囲には、NIS、 NIS+、 LDAPなどのネームサービスによってサービスが提供されるすべてのホストが含まれます。どのネームサービスが優先されるかは、/etc/nsswitch.confファイルでデータベースごとに設定されます。

- auth_attrエントリ - auth_attrデータベースにネームサービスの優先順位が設定されます。
- passwdエントリ - user_attrデータベースにネームサービスの優先順位が設定されます。
- prof_attrエントリ - prof_attrデータベースにネームサービスの優先順位が設定されます。また、exec_attrデータベースにもネームサービスの優先順位が設定されます。

  たとえば、セキュリティー属性を指定したコマンドを特定の権利プロファイルに割り当てた場合に、そのプロファイルが2つのネームサービスに存在するときは、最初のネームサービスのエントリだけが使用されます。

user_attrデータベース

user_attrデータベースには、ユーザーと役割の情報が格納されます。これらの情報は、passwdおよびshadowデータベースによって利用されます。user_attrデータベースには、承認、権利プロファイル、特権、割り当てられた役割など、さまざまなユーザー属性が格納されます。user_attrデータベースの各フィールドは次のようにコロンで区切ります。

```
user:qualifier:res1:res2:attr
```

フィールドの意味は次のとおりです。

user
: passwdデータベースに指定されているユーザー名または役割名。

qualifier:res1:res2
: これらのフィールドは、将来的な使用のために予約されています。

attr
セミコロン (;) で区切られた、鍵と値のペアからなるリスト (省略可能)。ユーザーがコマンドを実行したときに適用されるセキュリティー属性を表します。有効な鍵は、type、 auths、 profiles、 roles の 4 つです。

- type キーワードには、アカウントが通常ユーザーの場合は normal を設定します。役割の場合 type は role になります。
- auths キーワードには、auth_attr データベースの定義名から選択した承認名をコンマで区切って指定します。承認名には、ワイルドカードとしてアスタリスク (*) を使用できます。たとえば、solaris.device.* はすべての Oracle Solaris デバイスの承認を意味します。
- profiles キーワードには、prof_attr データベースに定義されている権利プロファイル名をコンマで区切って指定します。権利プロファイルの順序は、UNIX 検索パスと同様に動作します。実行するコマンドにどのセキュリティー属性が適用されるかは、そのコマンドが含まれているリストの最初のプロファイルによって決まります (属性を使用する場合)。
- roles キーワードは役割名をコンマで区切って指定します。役割も同じ user_attr データベースに定義されます。役割の場合は、type 値に role が設定されます。役割をほかの役割に割り当てることはできません。

次の例では、Operator 役割を標準的な user_attr データベースに定義する方法を示しています。また、Operator 役割をユーザー jdoe に割り当てる方法も示しています。役割とユーザーは、type キーワードによって識別されます。

```
% grep operator /etc/user_attr
jdoe::::type=normal;roles=operator
operator::::profiles=Operator;type=role
```

auth_attr データベース

承認はすべて auth_attr データベースに格納されます。承認は、ユーザー、役割、権利プロファイルに割り当てることができます。承認を権利プロファイルに配置し、権利プロファイルを役割のプロファイルリストに含めたあと、その役割をユーザーに割り当てることをお勧めします。

auth_attr データベースのフィールドは次のようにコロンで区切ります。

```
authname:res1:res2:short_desc:long_desc:attr
```

フィールドの意味は次のとおりです。

authname　承認を識別する一意の文字列。書式は *prefix*.[*suffix*]。Oracle Solaris では、承認の接頭辞として solaris を使用します。ほかのすべての承認には、承認を作成する組織のインターネットドメインを逆にしたもの

で始まる接頭辞を使用します (たとえば、com.xyzcompany)。接尾辞は、一般には機能領域と操作、およびどのように承認されるかを示します。

authnameが接頭辞と機能領域で構成され、ピリオドで終わるときは、GUI内でアプリケーションによって使用されるヘッダーとして機能します。2つの部分からなるauthnameは実際の承認ではありません。たとえば、authnameがsolaris.printmgr.の場合、ヘッダーとして使用されます。

authnameが「grant」で終わるときは、認可承認として機能します。認可承認を持つユーザーは、同じ接頭辞と機能領域で構成される承認をほかのユーザーに委託できます。たとえば、solaris.printmgr.grantがauthnameの場合は、認可承認として使用されます。solaris.printmgr.grantが許可されたユーザーは、solaris.printmgr.adminやsolaris.printmgr.nobannerなどの承認をほかのユーザーに委託する権利を持ちます。

res1:res2 これらのフィールドは、将来的な使用のために予約されています。

short_desc 承認の簡略名。この簡略名は、GUIのスクロールリストの中など、ユーザーインタフェースでの表示に適しています。

long_desc 詳しい記述。このフィールドには、承認の目的、承認が使用されるアプリケーション、承認を使用するユーザーの種類などを記述します。詳しい記述は、アプリケーションのヘルプテキストに表示できます。

attr 承認の属性を記述する鍵と値のペアをセミコロン (;) で区切ったリスト (オプション)。0または1つ以上の鍵を指定できます。

キーワードhelpにはHTML形式のヘルプファイルを指定します。ヘルプファイルは、/usr/lib/help/auths/locale/Cディレクトリのindex.htmlファイルからアクセスできます。

次の例は、標準的な値がいくつか設定されたauth_attrデータベースを示します。

```
% grep printer /etc/security/auth_attr
solaris.admin.printer.:::Printer Information::help=AuthPrinterHeader.html
solaris.admin.printer.delete:::Delete Printer Information::help=AuthPrinterDelete.html
solaris.admin.printer.modify:::Update Printer Information::help=AuthPrinterModify.html
solaris.admin.printer.read:::View Printer Information::help=AuthPrinterRead.html
```

solaris.admin.printer.はドット (.) で終わっているため、ヘッダーとして定義されます。ヘッダーは、承認の集合を作成するために、GUIによって使用されます。

prof_attr データベース

prof_attr データベースには、権利プロファイルに割り当てる名前、説明、ヘルプファイルの場所、特権、および承認が格納されます。権利プロファイルに割り当てられたコマンドとセキュリティー属性は、exec_attr データベースに格納されます。詳細については、255 ページの「exec_attr データベース」を参照してください。prof_attr データベースのフィールドは次のようにコロンで区切ります。

```
profname:res1:res2:desc:attr
```

フィールドの意味は次のとおりです。

profname　権利プロファイルの名前。権利プロファイル名では大文字と小文字が区別されます。この名前は、役割とユーザーにプロファイルを指定するために、user_attr データベースでも使用されます。

res1:res2　これらのフィールドは、将来的な使用のために予約されています。

desc　詳しい記述。このフィールドでは、権利プロファイルの使用に適したユーザーの種類など、権利プロファイルの目的を説明します。詳しい記述は、アプリケーションのヘルプテキストとして適している内容である必要があります。

attr　実行時にそのオブジェクトに適用するセキュリティー属性を記述する鍵と値のペアをセミコロン (;) で区切ったリスト (オプション)。0 または 1 つ以上の鍵を指定できます。有効な鍵は help です。profiles、 および auths。

キーワード help には HTML 形式のヘルプファイルを指定します。ヘルプファイルは、/usr/lib/help/profiles/locale/C ディレクトリの index.html ファイルからアクセスできます。

キーワード profiles は権利プロファイルをコンマで区切って指定します。これらのプロファイルは補助権利プロファイルと呼ばれます。

キーワード auths には、auth_attr データベースから選択した承認名をコンマで区切って指定します。承認名には、ワイルドカードとしてアスタリスク (*) を使用できます。

キーワード privs は特権をコンマで区切って指定します。これらの特権は、事実上、プロファイルシェルでのすべてのコマンドのためのものです。

次の例では、標準的な 2 つの prof_attr データベースエントリを示します。Printer Management 権利プロファイルは、Operator 権利プロファイルの補助権利プロファイルです。この例は、表示の都合上、折り返して記載されています。

```
% grep 'Printer Management' /etc/security/prof_attr
Printer Management:::                    Name of rights profile
Manage printers, daemons, spooling:      Description
help=RtPrntAdmin.html;                   Help file
auths=solaris.admin.printer.read,        Authorizations
solaris.admin.printer.modify,solaris.admin.printer.delete
...
Operator:::                              Name of rights profile
Can perform simple administrative tasks:    Description
profiles=Printer Management,             Supplementary rights profiles
Media Backup,All;
help=RtOperator.html                     Help file
```

exec_attrデータベース

exec_attr データベースでは、成功するためにセキュリティー属性を必要とするコマンドが定義されます。このコマンドは、権利プロファイルの一部です。セキュリティー属性を指定したコマンドは、プロファイルが割り当てられている役割またはユーザーが実行できます。

exec_attr データベースのフィールドは次のようにコロンで区切って指定します。

```
name:policy:type:res1:res2:id:attr
```

フィールドの意味は次のとおりです。

profname	権利プロファイルの名前。権利プロファイル名では大文字と小文字が区別されます。この名前は、prof_attr データベースのプロファイルを参照します。
policy	このエントリに関連付けるセキュリティーポリシー。現時点では、suser と solaris が有効なエントリです。solaris ポリシーでは、特権が認識されます。suser ポリシーでは、認識されません。
type	指定するエンティティーの種類。現在は、cmd (コマンド) が唯一の有効なエンティティーです。
res1:res2	これらのフィールドは、将来的な使用のために予約されています。
id	エンティティーを識別する文字列。コマンドには、完全パスかワイルドカード (*) をもつパスを指定します。引数を指定する場合は、引数をもつスクリプトを作成し、そのスクリプトを id に指定します。
attr	実行時にそのエンティティーに適用するセキュリティー属性を記述する鍵と値のペアをセミコロン (;) で区切ったリスト (オプション)。0 または 1 つ以上の鍵を指定できます。有効なキーワードのリストは、適用するポリシーによって異なります。

suserポリシーに対して有効な鍵は、euid、 uid、 egid、 gidの4つです。

- euidおよびuidキーワードには、1つのユーザー名またはユーザーID (UID) の数値が含まれます。euidを使用すると、コマンドは指定されたUIDで動作します。これは、実行可能ファイルにsetuidビットを設定することと同じです。uidを使用すると、コマンドは指定された実UIDと実効UIDで動作します。
- egidおよびgidキーワードには、1つのグループ名またはグループID (GID) の数値が含まれます。egidを使用すると、コマンドは指定されたGIDで動作します。これは、実行可能ファイルにsetgidビットを設定することと同じです。gidを使用すると、コマンドは指定された実GIDと実効GIDで動作します。

solarisポリシーに対して、有効なキーワードはprivsです。値は、コンマで区切られた特権のリストから構成されます。

次の例に、exec_attrデータベースの標準的な値をいくつか示します。

```
% grep 'File System Management' /etc/security/exec_attr
File System Management:suser:cmd:::/usr/sbin/ff:euid=0
File System Management:solaris:cmd:::/usr/sbin/mount:privs=sys_mount
...
```

policy.confファイル

policy.confファイルは、特定の権利プロファイル、特定の承認、および特定の特権をすべてのユーザーに与える方法を定義します。ファイル内の関連するエントリは、*key*=*value*のペアから構成されます。

- AUTHS_GRANTED=*authorizations* – 1つまたは複数の承認を示します。
- PROFS_GRANTED=*rights profiles* – 1つまたは複数の権利プロファイルを示します。
- PRIV_DEFAULT=*privileges* – 1つまたは複数の特権を示します。
- PRIV_LIMIT=*privileges* – すべての特権を示します。

次の例では、policy.confデータベースの標準的な値をいくつか示します。

```
# grep AUTHS /etc/security/policy
AUTHS_GRANTED=solaris.device.cdrw

# grep PROFS /etc/security/policy
PROFS_GRANTED=Basic Solaris User

# grep PRIV /etc/security/policy

#PRIV_DEFAULT=basic
#PRIV_LIMIT=all
```

権限の詳細は、199 ページの「特権 (概要)」を参照してください。

RBAC コマンド

この節では、RBAC の管理に使用するコマンドを一覧します。承認を使用してアクセス権を制御できるコマンドについても説明します。

RBAC を管理するコマンド

ローカルの RBAC データベースは手動で編集できますが、そのような編集はできるだけ避けてください。RBAC を使用したタスクへのアクセスを管理するために、次のコマンドが使用できます。

表 10–7　RBAC 管理コマンド

コマンドのマニュアルページ	説明
auths(1)	ユーザーに対する承認を表示します。
makedbm(1M)	dbm ファイルを作成します。
nscd(1M)	ネームサービスキャッシュデーモン。user_attr、 prof_attr、 および exec_attr データベースをキャッシュするときに使用します。svcadm コマンドを使用してデーモンを再起動します。
pam_roles(5)	PAM 用の役割アカウント管理モジュール。役割になる承認があるかを検査します。
pfexec(1)	プロファイルシェルによって使用されます。exec_attr データベースに指定されているセキュリティー属性を使用してコマンドを実行します。
policy.conf(4)	システムのセキュリティーポリシーの構成ファイル。与えられている承認、与えられている特権、およびその他のセキュリティー情報を一覧表示します。
profiles(1)	指定したユーザーの権利プロファイルを表示します。
roles(1)	指定したユーザーが引き受けられる役割を表示します。
roleadd(1M)	役割をローカルシステムに追加します。
roledel(1M)	役割をローカルシステムから削除します。
rolemod(1M)	ローカルシステム上の役割のプロパティーを変更します。
smattrpop(1M)	2 つのセキュリティー属性データベースをマージします。ローカルデータベースをネームサービスにマージするときに使用します。変換スクリプトを使用しないでアップグレードするときも使用します。

表 10–7 RBAC 管理コマンド (続き)

コマンドのマニュアルページ	説明
smexec(1M)	exec_attr データベースのエントリを管理します。認証を必要とします。
smmultiuser(1M)	ユーザーアカウントの一括操作を管理します。認証を必要とします。
smprofile(1M)	prof_attr および exec_attr データベースの権利プロファイルを管理します。認証を必要とします。
smrole(1M)	役割アカウントの役割とユーザーを管理します。認証を必要とします。
smuser(1M)	ユーザーのエントリを管理します。認証を必要とします。
useradd(1M)	ユーザーアカウントをシステムに追加します。ユーザーのアカウントに役割を割り当てるには、-R オプションを使用します。
userdel(1M)	ユーザーのログインをシステムから削除します。
usermod(1M)	システム上のユーザーのアカウントプロパティーを変更します。

承認を必要とするコマンド

次の表では、承認を使用して Oracle Solaris システムのコマンドオプションを制限する方法を示します。承認の詳細については、248 ページの「承認の命名と委託」を参照してください。

表 10–8 コマンドおよび関連する承認

コマンドのマニュアルページ	承認の要件
at(1)	solaris.jobs.user がすべてのオプションで必要です (at.allow ファイルおよび at.deny ファイルがない場合)
atq(1)	solaris.jobs.admin がすべてのオプションで必要です
cdrw(1)	solaris.device.cdrw がすべてのオプションで必要です。policy.conf ファイルにデフォルトで与えられます
crontab(1)	ジョブを送信するオプションの場合は、solaris.jobs.user が必要です (crontab.allow および crontab.deny ファイルがない場合) ほかのユーザーの crontab ファイルを一覧表示または変更する場合は、solaris.jobs.admin が必要です
allocate(1)	デバイスを割り当てる場合は、solaris.device.allocate (または、device_allocate ファイルに指定されている別承認) が必要です ほかのユーザーにデバイスを割り当てる場合 (-F オプション) は、solaris.device.revoke (または、 -device_allocate ファイルに指定されている別承認) が必要です

表 10–8 コマンドおよび関連する承認 (続き)

コマンドのマニュアルページ	承認の要件
deallocate(1)	ほかのユーザーのデバイスの割り当てを解除する場合は、solaris.device.allocate (または、device_allocate ファイルに指定されている別承認) が必要です 指定したデバイス (-F オプション) またはすべてのデバイス (-I オプション) の割り当てを強制的に解除する場合は、solaris.device.revoke (または、device_allocate に指定されている別承認) が必要です
list_devices(1)	ほかのユーザーのデバイスを一覧表示する場合 (U オプション) は、-solaris.device.revoke が必要です
sendmail(1M)	メールサブシステム機能にアクセスする場合は、solaris.mail が必要。メールキューを表示する場合は、solaris.mail.mailq が必要です

◆ ◆ ◆ 第 11 章

11 特権 (手順)

この章では、ご使用のシステムでの特権の管理と使用の手順について説明します。この章の内容は次のとおりです。

特権の概要については、199 ページの「特権 (概要)」を参照してください。参照情報については、第 12 章「特権 (参照)」を参照してください。

特権の管理と使用 (作業マップ)

次の作業マップは、特権を管理および使用するための作業マップです。

作業	説明	参照先
サイトで特権を使用します	特権の使用の割り当て、削除、追加、デバッグを含みます。	261 ページの「特権の管理 (作業マップ)」
コマンド実行時に特権を使用します	割り当てられた特権の使用を含みます。	270 ページの「特権の判断 (作業マップ)」

特権の管理 (作業マップ)

次の作業マップでは、特権の表示、特権の割り当て、および特権コマンドを含むスクリプトの実行の手順を示します。

作業	説明	参照先
プロセスに含まれる特権を判断します	プロセスに対する、有効な特権、継承可能な特権、許可された特権、および制限付きの特権のセットを一覧表示します。	262 ページの「プロセスの特権を判断する方法」
プロセスから漏れている特権を判断します	失敗したプロセスが必要とする特権を一覧表示します。	264 ページの「プログラムが必要とする特権を判断する方法」
特権をコマンドに追加します	特権を権利プロファイルのコマンドに追加します。ユーザーまたは役割を権利プロファイルに割り当てることができます。ユーザーは、プロファイルシェルで割り当てられた特権でコマンドを実行できるようになります。	266 ページの「特権をコマンドに追加する方法」
特権をユーザーに割り当てます	ユーザーまたは役割の継承可能な一連の特権を拡張します。この手順は、十分注意して実行してください。	266 ページの「特権をユーザーまたは役割に割り当てる方法」
ユーザーの特権を制限します	ユーザーの特権の基本セットを制限します。この手順は、十分注意して実行してください。	268 ページの「ユーザーまたは役割の特権を制限する方法」
特権付きのシェルスクリプトを実行します	特権をシェルスクリプトおよびシェルスクリプトのコマンドに追加します。その後、プロファイルシェルのスクリプトを実行します。	269 ページの「特権付きのコマンドを含むシェルスクリプトの実行方法」

特権の管理

ユーザーおよび役割の特権を管理する最も安全な方法として、権利プロファイルのコマンドに対して特権の使用を制限します。その後、権利プロファイルを役割に含めます。その役割をユーザーに割り当てます。ユーザーは、割り当てられた役割を引き受けると、特権付きコマンドをプロファイルシェルで実行できるようになります。次の手順は、特権の割り当て、特権の削除、および特権の使用のデバッグの方法を示したものです。

▼ プロセスの特権を判断する方法

この手順は、プロセスで使用可能な特権を判断する方法です。一覧には、特定のコマンドに割り当てられている特権は含まれません。

- シェルのプロセスで使用可能な特権を一覧表示します。

```
% ppriv pid
$ ppriv -v pid
```

pid プロセス番号です。二重ドル記号 (`$$`) を使用して親シェルのプロセス番号をコマンドに渡します。

`-v` 特権名の詳細な一覧表示を行います。

例 11–1 現在のシェルでの特権の判断

次の例では、ユーザーのシェルプロセスの親プロセスでの特権を一覧表示します。2 つ目の例では、特権の正式名を一覧表示します。出力の最初の文字は、次の特権セットを指しています。

E 有効な特権セットです。

I 継承可能な特権セットです。

P 許可された特権セットです。

L 制限付きの特権セットです。

```
% ppriv $$
1200:   -csh
flags = <none>
        E: basic
        I: basic
        P: basic
        L: all
% ppriv -v $$
1200:   -csh
flags = <none>
        E: file_link_any,net_access,proc_exec,proc_fork,proc_info,proc_session
        I: file_link_any,net_access,proc_exec,proc_fork,proc_info,proc_session
        P: file_link_any,net_access,proc_exec,proc_fork,proc_info,proc_session
        L: cpc_cpu,dtrace_kernel,dtrace_proc,dtrace_user,...,sys_time
```

例 11–2 引き受けることができる役割の特権の判断

役割では、管理シェルまたはプロファイルシェルが使用されます。役割に直接割り当てられている特権を一覧表示するには、役割を引き受け、役割のシェルを使用する必要があります。次の例では、役割 sysadmin に直接割り当てられている特権はありません。

```
% su - sysadmin
Password:       <Type sysadmin password>
$ /usr/ucb/whoami
sysadmin
$ ppriv -v $$
1400:   pfksh
flags = <none>
        E: file_link_any,net_access,proc_exec,proc_fork,proc_info,proc_session
        I: file_link_any,net_access,proc_exec,proc_fork,proc_info,proc_session
        P: file_link_any,net_access,proc_exec,proc_fork,proc_info,proc_session
        L: cpc_cpu,dtrace_kernel,dtrace_proc,dtrace_user,...,sys_time
```

▼ プログラムが必要とする特権を判断する方法

この手順では、コマンドまたはプロセスが必要とする特権を判断します。

始める前に　この手順は、コマンドまたはプロセスが失敗している場合に有効です。

1 **ppriv** デバッグコマンドに対する引数として失敗しているコマンドを入力します。

```
% ppriv -eD touch /etc/acct/yearly
touch[11365]: missing privilege "file_dac_write"
     (euid = 130, syscall = 224) needed at ufs_direnter_cm+0x27c
touch: /etc/acct/yearly cannot create
```

2 **/etc/name_to_sysnum** ファイルの **syscall** 番号を見つけ、失敗しているシステムコールを突き止めます。

```
% grep 224 /etc/name_to_sysnum
creat64                 224
```

例 11–3 特権の使用を検査するための truss コマンドの使用

truss コマンドは、通常のシェルで特権の使用をデバッグすることができます。たとえば、次のコマンドは、失敗した touch プロセスをデバッグします。

```
% truss -t creat touch /etc/acct/yearly
creat64("/etc/acct/yearly", 0666)
                      Err#13 EACCES [file_dac_write]
touch: /etc/acct/yearly cannot create
```

拡張された /proc インタフェースで、truss 出力のエラーコードのあとに欠如している特権がレポートされます。

例 11–4 プロファイルシェルで特権の使用を検査するための ppriv コマンドの使用

ppriv コマンドは、プロファイルシェルで特権の使用をデバッグすることができます。権利プロファイルをユーザーに割り当て、割り当てた権利プロファイルに特権付きのコマンドを含める場合、そのコマンドはプロファイルシェルで入力する必要があります。特権付きのコマンドを通常のシェルで入力すると、そのコマンドは特権では実行されません。

次の例で、jdoe ユーザーは、役割 objadmin を引き受けることができます。objadmin 役割には、Object Access Management 権利プロファイルが含まれます。この権利プロファイルによって、objadmin 役割は objadmin が所有しないファイルに関するアクセス権を変更することができます。

次の例で、jdoe は、useful.script ファイルに関するアクセス権を変更することができません。

```
jdoe% ls -l useful.script
-rw-r--r--  1 aloe  staff  2303 Apr 10 10:10 useful.script
jdoe% chown objadmin useful.script
chown: useful.script: Not owner
jdoe% ppriv -eD chown objadmin useful.script
chown[11444]: missing privilege "file_chown"
            (euid = 130, syscall = 16) needed at ufs_setattr+0x258
chown: useful.script: Not owner
```

jdoe が objadmin 役割を引き受けると、ファイルに関するアクセス権が変更されます。

```
jdoe% su - objadmin
Password:      <Type objadmin password>
$ ls -l useful.script
-rw-r--r--  1 aloe  staff  2303 Apr 10 10:10 useful.script
$ chown objadmin useful.script
$ ls -l useful.script
-rw-r--r--  1 objadmin  staff  2303 Apr 10 10:10 useful.script
$ chgrp admin useful.script
$ ls -l objadmin.script
-rw-r--r--  1 objadmin  admin  2303 Apr 10 10:11 useful.script
```

例 11–5 root ユーザーが所有するファイルの変更

この例では、特権エスカレーションに対する保護について説明します。詳細については、280 ページの「特権エスカレーションの防止」を参照してください。ファイルは、root ユーザーが所有します。権限の弱い objadmin 役割ではファイルの所有を変更するためにはすべての特権が必要なので、処理は失敗します。

```
jdoe% su - objadmin
Password:      <Type objadmin password>
$ cd /etc; ls -l system
-rw-r--r--  1 root  sys   1883 Oct 10 10:20 system
$ chown objadmin system
chown: system: Not owner
$ ppriv -eD chown objadmin system
chown[11481]: missing privilege "ALL"
     (euid = 101, syscall = 16) needed at ufs_setattr+0x258
chown: system: Not owner
```

▼ 特権をコマンドに追加する方法

コマンドを権利プロファイルに追加するときは、特権をコマンドに追加します。特権によって権利プロファイルを含む役割は管理コマンドを実行することができるようになりますが、ほかのスーパーユーザー機能は与えられません。

始める前に　コマンドまたはプログラムは、特権を認識できる必要があります。詳細については、205 ページの「プロセスが特権を取得する方法」を参照してください。

1. スーパーユーザーになるか、同等の役割を引き受けます。
   役割には、認証と特権コマンドが含まれます。役割の詳細については、210 ページの「RBAC の構成 (作業マップ)」を参照してください。
2. **Solaris 管理コンソールの GUI を開きます。**
   手順については、228 ページの「Solaris 管理コンソールで役割を引き受ける方法」を参照してください。
3. 権利ツールを使用して、該当するプロファイルを更新します。
   含めるコマンドを選択します。含めるコマンドごとに、コマンドが必要とする特権を追加します。

注意 – 権利プロファイルにコマンドを含め、含めたコマンドに特権を追加すると、コマンドは、プロファイルシェルで実行されたときに、それらの特権で実行されます。

プロファイルの順序は重要です。プロファイルシェルで、コマンドまたはアクションは、アカウントのプロファイルリストの最初のプロファイルで指定されたセキュリティー属性で実行されます。たとえば、`chgrp` コマンドが特権付きの Object Access Management 権利プロファイルに含まれていて、Object Access Management が `chgrp` が検出される最初のプロファイルである場合、`chgrp` コマンドは Object Access Management プロファイルで指定された特権で実行されます。

▼ 特権をユーザーまたは役割に割り当てる方法

特定の特権を持つ何人かのユーザーを常に信頼する場合があります。システムのごく一部に影響を与える非常に限定的な特権をユーザーに割り当てることをお勧めします。特権を直接割り当てることによる影響の詳細については、198 ページの「セキュリティー属性を直接割り当てる場合に考慮すべきセキュリティー事項」を参照してください。

次の手順により、ユーザー `jdoe` は高分解能タイマーを使用できるようになります。

1. **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

   Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2. 高分解能時間に影響を与える特権を、ユーザーの最初の継承可能な特権セットに追加します。

   ```
   $ usermod -K defaultpriv=basic,proc_clock_highres jdoe
   ```

   既存の値が `defaultpriv` キーワードの値で置き換えられます。このため、ユーザーが `basic` 特権を保持するために、値 `basic` を指定する必要があります。デフォルトの構成では、すべてのユーザーが基本特権を保持します。

3. 結果として生じる **user_attr** エントリを確認します。

   ```
   $ grep jdoe /etc/user_attr
   jdoe::::type=normal;defaultpriv=basic,proc_clock_highres
   ```

例 11–6 システム時間を構成するための特権付き役割の作成

この例では、システム上の時間の処理を唯一のタスクとする役割を作成します。

```
$ /usr/sadm/bin/smrole -D nisplus:/examplehost/example.domain \
-r primaryadm -l <Type primaryadm password> \
add -- -n clockmgr \
-c "Role that sets system time" \
-F "Clock Manager" \
-s /bin/pfksh \
-u 108 \
-P <Type clockmgr password> \
-K defaultpriv=basic,proc_priocntl,sys_cpu_config,
proc_clock_highres,sys_time
```

`-K` 行は、表示の都合上、折り返して記載されています。

役割がローカルに作成される場合、その役割に対する `user_attr` エントリは次のようになります。

```
clockmgr:::Role that sets system time:
type=role;defaultpriv=basic,proc_priocntl,sys_cpu_config,
proc_clock_highres,sys_time
```

▼ ユーザーまたは役割の特権を制限する方法

基本セットを削減するか、制限セットを削減することにより、ユーザーまたは役割が使用可能な特権を制限することができます。このような制限は意図しない面に影響を与える可能性があるため、この方法でユーザーの特権を制限するにはそれ相応の理由が必要です。

注意 – ユーザーに対して基本セットまたは制限セットが変更されたユーザーの機能を入念にテストしてください。

- 基本セットがデフォルトより少ないと、ユーザーはシステムを使用できない可能性があります。
- 制限セットがすべての特権より少ないと、実効 `UID=0` で実行する必要があるプロセスが失敗する可能性があります。

1. **ユーザーの基本セットおよび制限セットの特権を判断します。**

   手順については、262 ページの「プロセスの特権を判断する方法」を参照してください。

2. **(省略可能) 基本セットから特権の 1 つを削除します。**

   ```
   $ usermod -K defaultpriv=basic,!priv-name username
   ```

   `proc_session` 特権を削除することにより、ユーザーは現在のセッション外のプロセスを検査できなくなります。`file_link_any` 特権を削除することにより、ユーザーは所有していないファイルへのハードリンクを作成できなくなります。

注意 – `proc_fork` 特権または `proc_exec` 特権は削除しないでください。これらの特権がないと、ユーザーはシステムを使用することができません。実際のところ、これらの 2 つの特権は、ほかのプロセスに対して `fork()` または `exec()` すべきでないデーモンからそれ相応の理由がある場合にのみ削除されます。

3. **(省略可能) 制限セットから特権の 1 つを削除します。**

   ```
   $ usermod -K limitpriv=all,!priv-name username
   ```

4. ***username* の機能をテストします。**

   username としてログインし、*username* がシステム上で実行する必要のあるタスクを実行してみます。

例 11–7 ユーザーの制限セットからの特権の削除

次の例では、`jdoe` の最初のログインから開始されるすべてのセッションで `sys_linkdir` 特権を使用できないようにします。すなわち、ユーザーは、`su` コマンド

を実行したあとでも、ディレクトリへのハードリンクを作成することやディレクトリへのリンクを解除することができません。

```
$ usermod -K limitpriv=all,!sys_linkdir jdoe
$ grep jdoe /etc/user_attr
jdoe::::type=normal;defaultpriv=basic;limitpriv=all,!sys_linkdir
```

例 11–8 ユーザーの基本セットからの特権の削除

次の例では、jdoe の最初のログインから開始されるすべてのセッションで proc_session 特権を使用できないようにします。すなわち、ユーザーは、su コマンドを実行したあとでも、ユーザーのセッション外のいずれのプロセスも検査できません。

```
$ usermod -K defaultpriv=basic,!proc_session jdoe

$ grep jdoe /etc/user_attr
jdoe::::type=normal;defaultpriv=basic,!proc_session;limitpriv=all
```

▼ 特権付きのコマンドを含むシェルスクリプトの実行方法

注 – 継承された特権を持つコマンドを実行するシェルスクリプトを作成するときは、割り当てる特権付きのコマンドを該当する権利プロファイルに含める必要があります。

---

1. 1 行目は、スクリプトを **/bin/pfsh** またはほかのプロファイルシェルで開始します。

   ```
   #!/bin/pfsh
   # Copyright (c) 2009, 2011 by Oracle Corporation
   ```

2. スクリプトのコマンドが必要とする特権を判断します。

   ```
   % ppriv -eD script-full-path
   ```

3. Solaris 管理コンソールの GUI を開きます。

   手順については、228 ページの「Solaris 管理コンソールで役割を引き受ける方法」を参照してください。Primary Administrator など、権利プロファイルを作成することができる役割を選択します。

4. 権利ツールを使用して、該当するプロファイルを作成または更新します。

   スクリプトを選択し、実行に特権が必要なシェルスクリプトのコマンドをそれぞれ権利プロファイルに含めます。含めるコマンドごとに、コマンドが必要とする特権を追加します。

注意 – 権利プロファイルの順序は重要です。プロファイルシェルは、プロファイルリストのコマンドの最初のインスタンスを実行します。たとえば、chgrp コマンドが Object Access Management 権利プロファイルに含まれていて、Object Access Management が chgrp が検出される最初のプロファイルである場合、chgrp コマンドは Object Access Management プロファイルで指定された特権で実行されます。

5 権利プロファイルを役割に追加し、その役割をユーザーに割り当てます。

ユーザーは、プロファイルを実行するために、その役割を引き受け、役割のプロファイルシェルでスクリプトを実行します。

特権の判断 (作業マップ)

次の作業マップでは、割り当てられた特権を使用する手順を示します。

作業	説明	参照先
任意のシェルでユーザーとしての特権を表示します	直接割り当てられた特権を示します。プロセスのすべてはこれらの特権で実行されます。	270 ページの「直接割り当てられた特権を判断する方法」
特権で実行できるコマンドを判断します	権利プロファイルで特権が実行可能プログラムに割り当てられているとき、その実行可能プログラムはプロファイルシェルで入力する必要があります。	272 ページの「実行可能な特権付きコマンドを判断する方法」
役割が特権で実行できるコマンドを判断します	役割が特権で実行できるコマンドを判断する役割を引き受けます。	273 ページの「役割が実行可能な特権付きコマンドを判断する方法」

割り当てられた特権の判断

ユーザーに特権が直接割り当てられるとき、その特権はすべてのシェルで有効になります。ユーザーに特権が直接割り当てられないとき、ユーザーはプロファイルシェルを開く必要があります。たとえば、特権が割り当てられているコマンドがユーザーの権利プロファイルのリスト内の権利プロファイルに含まれるとき、ユーザーはプロファイルシェルでコマンドを実行する必要があります。

▼ 直接割り当てられた特権を判断する方法

次の手順は、特権が直接割り当てられたかどうかを判断する方法です。

注意 – 直接割り当てられた特権を不適切に使用すると、無意識のうちにセキュリティーを侵害する可能性があります。詳細については、198 ページの「セキュリティー属性を直接割り当てる場合に考慮すべきセキュリティー事項」を参照してください。

1 プロセスが使用可能な特権を一覧表示します。

手順については、262 ページの「プロセスの特権を判断する方法」を参照してください。

2 任意のシェルでアクションを起動し、コマンドを実行します。

有効なセットに一覧表示されている特権は、セッション全体を通して有効です。基本セットに加えて特権を直接割り当てた場合、その特権は有効なセットに一覧表示されます。

例 11–9 直接割り当てられた特権の判断

直接特権が割り当てられた場合、基本セットにはデフォルトの基本セットより多くの特権が含まれます。この例では、ユーザーは常時 `proc_clock_highres` 特権にアクセスできます。

```
% /usr/ucb/whoami
jdoe
% ppriv -v $$
1800:   pfksh
flags = <none>
        E: file_link_any,...,proc_clock_highres,proc_session
        I: file_link_any,...,proc_clock_highres,proc_session
        P: file_link_any,...,proc_clock_highres,proc_session
        L: cpc_cpu,dtrace_kernel,dtrace_proc,dtrace_user,...,sys_time
% ppriv -vl proc_clock_highres
        Allows a process to use high resolution timers.
```

例 11–10 役割に直接割り当てられた特権の判断

役割では、管理シェルまたはプロファイルシェルが使用されます。役割を引き受けるユーザーは、役割のシェルを使用して、役割に直接割り当てられている特権を一覧表示することができます。次の例では、役割 `realtime` に日時のプログラムを処理する特権が直接割り当てられます。

```
% su - realtime
Password:       <Type realtime password>
$ /usr/ucb/whoami
realtime
$ ppriv -v $$
1600:   pfksh
flags = <none>
```

```
E: file_link_any,...,proc_clock_highres,proc_session,sys_time
I: file_link_any,...,proc_clock_highres,proc_session,sys_time
P: file_link_any,...,proc_clock_highres,proc_session,sys_time
L: cpc_cpu,dtrace_kernel,dtrace_proc,dtrace_user,...,sys_time
```

▼ 実行可能な特権付きコマンドを判断する方法

ユーザーに直接特権が割り当てられていないとき、ユーザーは権利プロファイルによって特権付きコマンドにアクセスすることができます。権利プロファイルのコマンドは、プロファイルシェルで実行する必要があります。

始める前に　Solaris 管理コンソールに対して認証を行うユーザーまたは役割は、`solaris.admin.usermgr.read` 承認を得る必要があります。Basic Solaris User 権利プロファイルには、この承認が含まれます。

1 割り当てられている権利プロファイルを判断します。

```
$ /usr/sadm/bin/smuser list -- -n username -l

Authenticating as user: admin
... Please enter a string value for: password ::
...
User name:      username
User ID (UID):  130
Primary group:  staff
Secondary groups:
Comment: object mgt jobs
Login Shell: /bin/sh
Home dir server: system
Home directory: /export/home/username
AutoHome setup: True
Mail server: system
Rights: Object Access Management
Assigned Roles:
```

2 「**Rights:**」で始まる行を探します。

「Rights」行には、直接割り当てた権利プロファイルの名前が一覧表示されます。

3 **exec_attr** データベースの権利プロファイルの名前を見つけます。

```
$ cd /etc/security
$ grep "Object Access Management" exec_attr
Object Access Management:solaris:cmd:::/usr/bin/chgrp:privs=file_chown
Object Access Management:solaris:cmd:::/usr/bin/chown:privs=file_chown
Object Access Management:suser:cmd:::/usr/bin/chgrp:euid=0
Object Access Management:suser:cmd:::/usr/bin/chmod:euid=0
...
```

特権が追加されたコマンドは、`solaris` ポリシーエントリの最後に一覧表示されます。

4 プロファイルシェルで、特権を必要とするコマンドを入力します。

コマンドを通常のシェルで入力すると、そのコマンドは特権で実行されず、失敗します。

```
% pfsh
$
```

例 11–11 プロファイルシェルでの特権付きコマンドの実行

次の例で、ユーザー jdoe は、通常のシェルからはファイルに関するグループアクセス権を変更できません。しかし、jdoe は、プロファイルシェルでコマンドを入力すると、グループアクセス権を変更できます。

```
% whoami
jdoe
% ls -l useful.script
-rwxr-xr-- 1 nodoe eng 262 Apr 2 10:52 useful.script
chgrp staff useful.script
chgrp: useful.script: Not owner
% pfksh
$ /usr/ucb/whoami
jdoe
$ chgrp staff useful.script
$ chown jdoe useful.script
$ ls -l useful.script
-rwxr-xr-- 1 jdoe staff 262 Apr 2 10:53 useful.script
```

▼ 役割が実行可能な特権付きコマンドを判断する方法

役割は、特権が割り当てられたコマンドを含む権利プロファイルによって、特権付きコマンドにアクセスすることができます。特権付きコマンドに対するアクセス権をユーザーに与える最も安全な方法は、役割をコマンドに割り当てることです。ユーザーは、役割を引き受けると、その役割の権利プロファイルに含まれるすべての特権付きコマンドを実行することができます。

始める前に Solaris 管理コンソールに対して認証を行うユーザーまたは役割は、solaris.admin.usermgr.read 承認を得る必要があります。Basic Solaris User 権利プロファイルには、この承認が含まれます。

1 引き受けることができる役割を判断します。

```
$ /usr/sadm/bin/smuser list -- -n username -l
Authenticating as user: primadmin
...
User name:      username
User ID (UID):  110
Primary group:  staff
```

```
Secondary groups:
Comment: Has admin roles
Login Shell: /bin/sh
...
Rights:
Assigned Roles: primadmin, admin
```

2 「**Assigned Roles:**」で始まる行を探します。

「Assigned Roles」行には、引き受けることができる役割が一覧表示されます。

3 役割の 1 つに含まれる権利プロファイルを判断します。

```
% su - devadmin
Enter password:       Type devadmin password
$ whoami
devadmin
$ profiles
Device Security

$ /usr/sadm/bin/smuser list -- -n admin -l
Authenticating as user: primadmin
...
User name:      admin
User ID (UID):  101
Primary group:  sysadmin
Secondary groups:
Comment: system administrator
Login Shell: /bin/pfksh
...
Rights: System Administrator
Assigned Roles:
```

4 「**Rights:**」行で役割の権利プロファイルの名前を探します。

5 **prof_attr** データベースの権利プロファイルを見つけます。

System Administrator プロファイルはプロファイルの集合なので、System Administrator プロファイルのプロファイルを一覧表示する必要があります。

```
$ cd /etc/security
$ grep "System Administrator" prof_attr
System Administrator:::Can perform most non-security administrative
tasks:profiles=Audit Review,Printer Management,Cron Management,
Device Management,File System Management,Mail Management,Maintenance
and Repair,Media Backup,Media Restore,Name Service Management,Network
Management,Object Access Management,Process Management,Software
Installation,User Management,All;help=RtSysAdmin.html
```

6 権利プロファイルごとに、**exec_attr** データベースの権利プロファイルを見つけます。

たとえば、Network Management プロファイルは、System Administrator プロファイルの補助プロファイルです。Network Management プロファイルには、多数の特権付きコマンドが含まれます。

```
$ cd /etc/security
$ grep "Network Management" exec_attr
Network Management:solaris:cmd:::/usr/sbin/ifconfig:privs=sys_net_config
Network Management:solaris:cmd:::/usr/sbin/route:privs=sys_net_config
...
```

コマンドおよびそれらに割り当てられた特権は、`solaris` ポリシーエントリの最後の 2 つのフィールドです。役割のプロファイルシェルで、これらのコマンドを実行することができます。

例 11–12 役割での特権付きコマンドの実行

ユーザーが役割を引き受けると、シェルはプロファイルシェルになります。したがって、コマンドは、コマンドに割り当てられた特権で実行されます。次の例で、`admin` 役割は、 `useful.script` ファイルに関する権利を変更することができます。

```
% whoami
jdoe
% ls -l useful.script
-rwxr-xr-- 1 elsee eng 262 Apr 2 10:52 useful.script
chgrp admin useful.script
chgrp: useful.script: Not owner
% su - admin
Password:      <Type admin password>
$ /usr/ucb/whoami
admin
$ chgrp admin useful.script
$ chown admin useful.script
$ ls -l useful.script
-rwxr-xr-- 1 admin admin 262 Apr 2 10:53 useful.script
```

◆ ◆ ◆ 第 12 章

12 特権 (参照)

この章の内容は次のとおりです。

特権の使用方法については、第 11 章「特権 (手順)」を参照してください。概要については、199 ページの「特権 (概要)」を参照してください。

特権を扱うための管理コマンド

次の表では、特権を扱うために使用可能なコマンドを一覧表示します。

表 12–1 特権を扱うためのコマンド

目的	コマンド	マニュアルページ
プロセスの特権を検査します	`ppriv -v` *pid*	ppriv(1)
プロセスの特権を設定します	`ppriv -s` *spec*	
システム上の特権を一覧表示します	`ppriv -l`	
特権とその説明を一覧表示します	`ppriv -lv` *priv*	
特権の障害をデバッグします	`ppriv -eD` *failed-operation*	
特権を新しいローカルユーザーに割り当てます	`useradd`	useradd(1M)
特権を既存のローカルユーザーに追加します	`usermod`	usermod(1M)

表 12–1 特権を扱うためのコマンド (続き)

目的	コマンド	マニュアルページ
特権をネームサービスのユーザーに割り当てます	`smuser`	`smuser`(1M)
特権を新しいローカル役割に割り当てます	`roleadd`	`roleadd`(1M)
特権を既存のローカル役割に追加します	`rolemod`	`rolemod`(1M)
特権をネームサービスの役割に割り当てます	`smrole`	`smrole`(1M)
デバイスポリシーを表示します	`getdevpolicy`	`getdevpolicy`(1M)
デバイスポリシーを設定します	`devfsadm`	`devfsadm`(1M)
オープンデバイスでのデバイスポリシーを更新します	`update_drv -p` *policy driver*	`update_drv`(1M)
デバイスポリシーをデバイスに追加します	`add_drv -p` *policy driver*	`add_drv`(1M)

コマンド、ユーザー、および役割への特権の割り当ては、Solaris 管理コンソールの GUI で行うことをお勧めします。詳細については、228 ページの「Solaris 管理コンソールで役割を引き受ける方法」を参照してください。

特権情報が含まれるファイル

次のファイルには、特権に関する情報が含まれます。

表 12–2 特権情報が含まれるファイル

ファイルおよびマニュアルページ	キーワード	説明
`/etc/security/policy.conf` `policy.conf`(4)	`PRIV_DEFAULT`	システムに対する特権の継承可能なセット
	`PRIV_LIMIT`	システムに対する特権の制限セット

表 12–2 特権情報が含まれるファイル (続き)

ファイルおよびマニュアルページ	キーワード	説明
/etc/user_attr user_attr(4)	ユーザーまたは役割のエントリでの privs キーワード ユーザーまたは役割のエントリでの defaultpriv キーワード 値は通常、Solaris 管理コンソールの GUI で設定します	ユーザーまたは役割に対する特権の継承可能なセット
	ユーザーまたは役割のエントリでの limitpriv キーワード 値は通常、Solaris 管理コンソールの GUI で設定します	ユーザーまたは役割に対する特権の制限セット
/etc/security/exec_attr exec_attr(4)	プロファイルのコマンドに対するエントリでの privs キーワード コマンドに対するポリシーは solaris である必要があります	権利プロファイルのコマンドに割り当てられた特権の一覧
syslog.conf syslog.conf(4)	デバッグメッセージ用のシステムログファイル priv.debug エントリで設定されるパス	特権デバッグログ

注 – exec_attr および user_attr データベースは直接編集しないでください。特権を管理するには、Solaris 管理コンソールまたは smuser などのコマンドを使用します。詳細は、smc(1M) および smuser(1M) のマニュアルページを参照してください。手順については、261 ページの「特権の管理 (作業マップ)」を参照してください。

特権と監査

特権の使用は監査することができます。プロセスで特権が使用される場合は常に、upriv 監査トークン内の監査トレールに特権の使用が記録されます。特権の名前がレコードに含まれる場合、テキスト形式が使用されます。次の監査イベントにより、特権の使用が記録されます。

- AUE_SETPPRIV 監査イベント – 特権セットが変更されたときに監査記録を生成します。AUE_SETPPRIV 監査イベントは pm クラスにあります。
- AUE_MODALLOCPRIV 監査イベント – カーネル外から特権が追加されたときに監査記録を生成します。AUE_MODALLOCPRIV 監査イベントは ad クラスにあります。
- AUE_MODDEVPLCY 監査イベント – デバイスポリシーが変更されたときに監査記録を生成します。AUE_MODDEVPLCY 監査イベントは ad クラスにあります。

- AUE_prof_cmd 監査イベント – コマンドがプロファイルシェルで実行されたときに監査記録を生成します。AUE_prof_cmd 監査イベントは as および ua 監査クラスにあります。特権の名前は、監査レコードに含まれます。

基本セットに含まれる特権が正常に使用される場合は、監査されません。ユーザーの基本セットから削除された基本特権の使用を試みる場合、監査されます。

特権エスカレーションの防止

Oracle Solaris カーネルは特権エスカレーションを防ぎます。特権エスカレーションとは、特権によってプロセスがその範囲を越えたことを行えるようになることです。プロセスが特権の範囲を超えることを防ぐために、無防備なシステム変更を行うには特権の完全セットが必要です。たとえば、root (UID=0) が所有するファイルまたはプロセスは、特権の完全セットを備えたプロセスによってのみ変更できます。root アカウントは、特権がなくても root が所有するファイルを変更することができます。しかし、root ユーザー以外は、root が所有するファイルを変更するにはすべての特権が必要です。

同様に、デバイスへのアクセスを提供する操作には、有効なセットのすべての特権が必要です。

file_chown_self および proc_owner は、特権エスカレーションが生じやすい特権です。file_chown_self は、プロセスがそのファイルを渡せるようにする特権です。proc_owner は、プロセス自身が所有しないプロセスを調査できるようにする特権です。

file_chown_self 特権は、rstchown システム変数によって制限されます。rstchown 変数が 0 に設定されると、file_chown_self 特権は、システムおよび全ユーザーの初期の継承可能セットから削除されます。rstchown システム変数の詳細については、chown(1) のマニュアルページを参照してください。

file_chown_self 特権のコマンドへの割り当て、プロファイルへの配置、およびプロファイルシェルで使用するための役割への割り当ては最も安全に行います。

proc_owner 特権は、プロセス UID を 0 にするには十分ではありません。任意の UID のプロセスを UID=0 にするには、すべての特権が必要です。proc_owner 特権はシステム上のすべてのファイルに無制限の読み取りアクセス権を与えるので、この特権のコマンドへの割り当て、プロファイルへの配置、およびプロファイルシェルで使用するための役割への割り当ては最も安全に行います。

注意 – `file_chown_self` 特権または `proc_owner` 特権がユーザーの初期の継承可能セットに含まれるようにユーザーのアカウントを変更することができます。これらの強力な特権を任意のユーザー、役割、またはシステムに対する特権の継承可能セットに配置するには、セキュリティー上の相応の理由がなければなりません。

デバイスに対する特権エスカレーションを防ぐ方法の詳細については、207 ページの「特権とデバイス」を参照してください。

レガシーアプリケーションと特権モデル

レガシーアプリケーションに対応するために、特権の実装はスーパーユーザーモデルと特権モデルで動作します。カーネルは、プログラムが特権で動作するように設計されていること示す `PRIV_AWARE` フラグを自動的に追跡します。特権を認識しない子プロセスについて検討してください。親プロセスから継承された特権はどれも、子の許可されたセットおよび有効なセットで使用可能です。子プロセスで UID が 0 に設定されていると、子プロセスが完全なスーパーユーザー機能を持たない場合があります。プロセスの有効なセットおよび許可されたセットは、子の制限セットの特権に限定されます。このように、特権を認識するプロセスの制限セットによって、特権を認識しない子プロセス root 特権が制限されます。

パート IV

暗号化サービス

ここでは、Oracle Solaris OS が提供する集中型の暗号化および公開鍵技術サービスについて説明します。

◆ ◆ ◆ 第 13 章

13 Oracle Solaris の暗号化フレームワーク (概要)

この章では、Oracle Solaris の暗号化フレームワークについて説明します。この章の内容は次のとおりです。

Oracle Solaris の暗号化フレームワークの管理方法と使用方法については、第 14 章「Oracle Solaris の暗号化フレームワーク (手順)」を参照してください。

Oracle Solaris の暗号化フレームワークの新機能

Solaris 10 1/06: フレームワークの `libpkcs11.so` ライブラリには、「メタスロット」という新しいコンポーネントが含まれています。メタスロットは、フレームワークにインストールされているすべてのトークンとスロットの機能を結合させて単一の仮想スロットで提供するコンポーネントです。メタスロットにより、事実上、アプリケーションから利用可能なすべての暗号化サービスに単一のスロットを通じて透過的に接続できるようになります。

- 詳細は、287 ページの「Oracle Solaris の暗号化フレームワークの用語」にあるスロット、メタスロット、およびトークンの定義を参照してください。
- メタスロットの管理方法については、`cryptoadm(1M)` のマニュアルページを参照してください。
- Oracle Solaris の新機能の完全な一覧や各 Oracle Solaris リリースの説明については、『Oracle Solaris 10 8/11 の新機能』を参照してください。

Oracle Solaris の暗号化フレームワーク

Oracle Solaris の暗号化フレームワークは、暗号化要求を処理するアルゴリズムと PKCS #11 ライブラリの共通の格納場所を提供します。PKCS #11 ライブラリは、RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki) に従って実装されます。

暗号化フレームワークは、現在、Kerberos および IPsec に対する暗号化要求をカーネルレベルで処理します。ユーザーレベルコンシューマは、`libsasl` や IKE などです。

米国の輸出法では、公開された暗号化インタフェースは使用の制限が義務付けられています。Oracle Solaris の暗号化フレームワークは、カーネル暗号化プロバイダおよび PKCS #11 暗号化プロバイダの署名を義務付けることにより、この現行法を満たしています。詳細については、290 ページの「Sun 以外のソフトウェアのためのバイナリ署名」を参照してください。

暗号化フレームワークにより、暗号化サービスのプロバイダは、そのサービスが Oracle Solaris OS の多数のコンシューマに使用されるようにすることができます。プロバイダはプラグインとも言います。暗号化フレームワークでは、3 種類のプラグインが使用可能です。

- ユーザーレベルプラグイン – `pkcs11_softtoken.so.1` など、PKCS #11 ライブラリを使用することによってサービスを提供する共有オブジェクト。
- カーネルレベルプラグイン – AES など、ソフトウェアの暗号化アルゴリズムの実装を提供するカーネルモジュール。

  暗号化フレームワークのアルゴリズムの多くは、SSE2 命令セットを備えた x86 および SPARC ハードウェア用に最適化されます。
- ハードウェアプラグイン – デバイスドライバおよび関連するハードウェアアクセラレータ。たとえば、Niagara チップ、ncp および n2cp デバイスドライバです。ハードウェアアクセラレータにより、オペレーティングシステムの高価な暗号化機能が肩代わりされます。たとえば、Sun Crypto Accelerator 6000 ボードがあります。

暗号化フレームワークは、ユーザーレベルプロバイダ用に標準インタフェースとして PKCS #11, v2.11 ライブラリを実装しています。このライブラリは、Sun 以外のアプリケーションがプロバイダに到達するために使用することができます。サードパーティーは、署名付きライブラリ、署名付きカーネルアルゴリズムモジュール、および署名付きデバイスドライバを暗号化フレームワークに追加することもできます。これらのプラグインは、`pkgadd` ユーティリティーによって Sun 以外のソフトウェアがインストールされると追加されます。暗号化フレームワークの主なコンポーネントの一覧図については、『Oracle Solaris セキュリティーサービス開発ガイド』の第 8 章「Oracle Solaris 暗号化フレームワークの紹介」を参照してください。

Oracle Solaris の暗号化フレームワークの用語

次の定義と例のリストは、暗号化フレームワークでの作業時に役立ちます。

- アルゴリズム – 暗号化のアルゴリズム。入力を暗号化またはハッシュする、確立した再帰的な計算手順です。暗号化アルゴリズムには対称と非対称があります。対称アルゴリズムでは、暗号化と復号化に同じ鍵が使用されます。非対称アルゴリズムは、公開鍵暗号化で使用され、2 つの鍵を必要とします。ハッシングもアルゴリズムです。

  アルゴリズムの例として、次のものがあります。

  - AES や ARCFOUR などの対称アルゴリズム
  - Diffie-Hellman や RSA などの非対称アルゴリズム
  - MD5 などのハッシング機能
- コンシューマ – プロバイダから提供される暗号化サービスのユーザーです。コンシューマになりえるものとして、アプリケーション、エンドユーザー、カーネル処理などが挙げられます。

  コンシューマの例として、次のものがあります。

  - IKE などのアプリケーション
  - `encrypt` コマンドを実行する通常のユーザーなどのエンドユーザー
  - IPsec などのカーネル操作
- メカニズム – 特定の目的のアルゴリズムのモードのアプリケーションです。

  たとえば、認証に適用される DES メカニズム (CKM_DES_MAC など) は、暗号化に適用されるメカニズム (CKM_DES_CBC_PAD) とは別です。
- メタスロット – フレームワークに読み込まれているほかのスロットの機能を単一のスロットに統合したものです。メタスロットは、フレームワークを通じて利用可能なプロバイダのすべての機能を取り扱う際の作業を軽減します。メタスロットを使用するアプリケーションから処理要求を受けると、実際のスロットのうちどのスロットがその処理を行うべきかをメタスロットが判断します。メタスロットの機能は構成可能ですが、構成が必須ではありません。メタスロットはデフォルトでは有効になっています。メタスロットの構成方法については、`cryptoadm(1M)` のマニュアルページを参照してください。
- モード – 暗号化アルゴリズムのバージョンです。たとえば、CBC (Cipher Block Chaining) は、ECB (Electronic Code Book) とは別のモードです。AES アルゴリズムには、CKM_AES_ECB と CKM_AES_CBC の 2 つのモードがあります。
- ポリシー – どのメカニズムを使用できるようにするかについて、管理者によって選択されるものです。デフォルトでは、すべてのプロバイダとすべてのメカニズムが使用可能です。メカニズムを無効にすることは、ポリシーの適用です。無効になっていたメカニズムを有効にすることも、ポリシーの適用です。

- プロバイダ – コンシューマが使用する暗号化サービスです。プロバイダは、フレームワークに組み込まれる（プラグインする）ので、プラグインとも呼ばれます。

  プロバイダの例として、次のものがあります。

  - `pkcs11_softtoken.so` などの PKCS #11 ライブラリ
  - `aes` や `arcfour` などの暗号化アルゴリズムのモジュール
  - Sun Crypto Accelerator 6000 の `mca` ドライバなど、デバイスドライバおよび関連するハードウェアアクセラレータ
- スロット – 1 つまたは複数の暗号化デバイスとのインタフェースです。各スロットは物理的なリーダー (読み取り器) またはその他のデバイスインタフェースに相当し、スロットにはトークンが 1 つ搭載されていることがあります。トークンは、フレームワーク内の暗号化デバイスを論理的に表示します。
- トークン – スロット内で、フレームワーク内の暗号化デバイスを論理的に表示するものです。

Oracle Solaris の暗号化フレームワークの適用範囲

暗号化フレームワークには、管理者、ユーザー、およびプロバイダを提供する開発者向けのコマンドが用意されています。

- 管理コマンド – `cryptoadm` コマンドには、使用可能なプロバイダとその機能を一覧表示する `list` サブコマンドが用意されています。通常のユーザーは、`cryptoadm list` コマンドおよび `cryptoadm --help` コマンドを実行することができます。

  それ以外の `cryptoadm` サブコマンドでは、Crypto Management 権利プロファイルを含む役割を引き受けるか、スーパーユーザーになる必要があります。`disable`、`install`、 および `uninstall` などのサブコマンドを使用して、暗号化フレームワークを管理できます。詳細は、`cryptoadm(1M)` のマニュアルページを参照してください。

  `svcadm` コマンドを使用して、`kcfd` デーモンの管理やカーネルの暗号化ポリシーの更新を行うことができます。詳細は、`svcadm(1M)` のマニュアルページを参照してください。
- ユーザーレベルコマンド – `digest` コマンドおよび `mac` コマンドによって、ファイル整合性サービスが提供されます。`encrypt` および `decrypt` コマンドは、ファイルが傍受されるのを防ぎます。これらのコマンドを使用する場合は、294 ページの「Oracle Solaris 暗号化フレームワークによるファイルの保護 (作業マップ)」を参照してください。
- **Sun** 以外のプロバイダのためのバイナリ署名 – `elfsign` コマンドによって、暗号化フレームワーク内での使用のためにサードパーティーがバイナリに署名することができます。フレームワークに追加可能なバイナリは、PKCS #11 ライブラリ、カーネルアルゴリズムモジュール、およびハードウェアデバイスドライバで

す。elfsign コマンドを使用する場合は、『Oracle Solaris セキュリティーサービス開発ガイド』の付録 F「暗号化プロバイダのパッケージ化と署名」を参照してください。

Oracle Solaris 暗号化フレームワークの管理コマンド

cryptoadm コマンドは、動作中の暗号化フレームワークを管理します。このコマンドは、Crypto Management 権利プロファイルの一部です。この権利プロファイルは、暗号化フレームワークを安全に管理する役割に割り当てることができます。cryptoadm コマンドは、次の内容を管理します。

- 暗号化プロバイダ情報の表示
- プロバイダメカニズムの無効化または有効化
- Solaris 10 1/06: メタスロットの無効化または有効化

svcadm コマンドは、暗号化サービスデーモン kcfd を有効化、更新、および無効化するために使用されます。このコマンドは、サービス管理機能 (SMF) の一部です。svc:/system/cryptosvcs は、暗号化フレームワークのサービスインスタンスです。詳細は、smf(5) および svcadm(1M) のマニュアルページを参照してください。

Oracle Solaris 暗号化フレームワークのユーザーレベルコマンド

Oracle Solaris の暗号化フレームワークでは、ファイルの整合性の確認、ファイルの暗号化、およびファイルの復号化を行うユーザーレベルコマンドが用意されています。独立したコマンド elfsign によって、フレームワークでの使用のためにプロバイダがバイナリに署名することができます。

- digest コマンド – 1 つまたは複数のファイルまたは標準入力のメッセージ要約を計算します。要約は、ファイルの整合性を検証するのに便利です。SHA1 および MD5 は、要約機能の例です。
- mac コマンド – 1 つまたは複数のファイルまたは標準入力のメッセージ認証コード (MAC) を計算します。MAC は、データを認証されたメッセージに関連付けます。MAC によって、受信者は、メッセージの送信者、およびメッセージが改ざんされていないことを検証できるようになります。sha1_mac メカニズムおよび md5_hmac メカニズムが MAC を計算します。
- encrypt コマンド – 対称暗号でファイルまたは stdin を暗号化します。encrypt -l コマンドは、使用可能なアルゴリズムを一覧表示します。ユーザーレベルライブラリで一覧表示されるメカニズムは、encrypt コマンドで使用可能です。暗号化フレームワークでは、ユーザーの暗号化のために AES、 DES、 3DES (Triple-DES)、 および ARCFOUR メカニズムが用意されています。

- `decrypt` コマンド – `encrypt` コマンドで暗号化されたファイルまたは stdin を復号化します。`decrypt` コマンドは、元のファイルの暗号化に使用されたのと同一の鍵とメカニズムを使用します。

Sun 以外のソフトウェアのためのバイナリ署名

`elfsign` コマンドは、Oracle Solaris暗号化フレームワークでの使用のためにプロバイダに署名する手段です。一般に、このコマンドはプロバイダの開発者によって実行されます。

`elfsign` コマンドには、Sun の証明書を要求するためのサブコマンドとバイナリ署名を行うためのサブコマンドが用意されています。署名を確認するサブコマンドもあります。署名されていないバイナリは、Oracle Solaris の暗号化フレームワークで使用することができません。1 つまたは複数のプロバイダに署名するには、Sun の証明書と、その証明書の要求に使用された非公開鍵が必要です。詳細は、『Oracle Solaris セキュリティーサービス開発ガイド』の付録 F「暗号化プロバイダのパッケージ化と署名」を参照してください。

Oracle Solaris の暗号化フレームワークのプラグイン

サードパーティーは、Oracle Solaris 暗号化フレームワークに自身のプロバイダをプラグインできます。Sun 以外のプロバイダとは、次のオブジェクトのいずれかです。

- PKCS #11 共有ライブラリ
- 暗号化アルゴリズム、MAC 機能、要約機能など、ロード可能なカーネルソフトウェアモジュール
- ハードウェアアクセラレータ用のカーネルデバイスドライバ

プロバイダのオブジェクトは、Sun の証明書付きで署名されている必要があります。証明書要求は、サードパーティーが選択する非公開鍵と Sun が提供する証明書に基づきます。証明書要求は Sun に送信され、サードパーティーが登録されたあと、証明書が発行されます。次にサードパーティーは Sun の証明書付きでそのプロバイダオブジェクトに署名します。

ロード可能なカーネルソフトウェアモジュールおよびハードウェアアクセラレータ用のカーネルデバイスドライバも、カーネルに登録する必要があります。登録は、Oracle Solaris 暗号化フレームワークの SPI (サービスプロバイダインタフェース) で行います。

プロバイダをインストールするために、サードパーティーは署名付きのオブジェクトと Sun の証明書をインストールするパッケージを提供します。パッケージには証明書が含まれ、管理者が証明書を安全なディレクトリに格納する必要があります。詳細は、『Oracle Solaris セキュリティーサービス開発ガイド』の付録 F「暗号化プロバイダのパッケージ化と署名」を参照してください。

暗号化サービスとゾーン

大域ゾーンと各非大域ゾーンには、それぞれの /system/cryptosvc サービスが用意されています。大域ゾーンで暗号化サービスが有効になったり更新されたりすると、大域ゾーンで kcfd デーモンが起動され、大域ゾーンに対するユーザーレベルポリシーが設定され、システムに対するカーネルポリシーが設定されます。非大域ゾーンでサービスが有効になったり更新されたりすると、その非大域ゾーンで kcfd デーモンが起動され、そのゾーンに対するユーザーレベルポリシーが設定されます。カーネルポリシーは、大域ゾーンによって設定されています。

ゾーンの詳細は、『Oracle Solaris のシステム管理 (Oracle Solaris コンテナ : 資源管理と Oracle Solaris ゾーン)』のパート II「ゾーン」を参照してください。永続的なアプリケーションを管理するサービス管理機能の詳細は、『Solaris のシステム管理 (基本編)』の第 18 章「サービスの管理 (概要)」および smf(5) のマニュアルページを参照してください。

◆ ◆ ◆ 第 14 章

Oracle Solaris の暗号化フレームワーク (手順)

この章では、Oracle Solaris の暗号化フレームワークの使用方法について説明します。この章の内容は次のとおりです。

暗号化フレームワークの使用 (作業マップ)

次の作業マップでは、暗号化フレームワークの使用についての作業を示します。

作業	説明	参照先
個々のファイルまたはファイルのセットを保護します	ファイルの内容が改ざんされていないことを確認します。ファイルが侵入者によって解読されるのを防ぎます。これらの手順は通常のユーザーが行うことができます。	294 ページの「Oracle Solaris 暗号化フレームワークによるファイルの保護 (作業マップ)」
暗号化フレームワークを管理します	ソフトウェアプロバイダを追加、構成、および削除します。ハードウェアプロバイダのメカニズムを無効および有効にします。これらの手順は管理上の手順です。	305 ページの「暗号化フレームワークの管理 (作業マップ)」
プロバイダに署名します	プロバイダが Oracle Solaris 暗号化フレームワークに追加されるようにします。これらの手順は開発者の手順です。	『Oracle Solaris セキュリティーサービス開発ガイド』の付録 F「暗号化プロバイダのパッケージ化と署名」

Oracle Solaris 暗号化フレームワークによるファイルの保護 (作業マップ)

Oracle Solaris 暗号化フレームワークは、ファイルの保護に役立ちます。次の作業マップでは、使用可能なアルゴリズムを一覧表示する手順、および暗号化によってファイルを保護する手順を示します。

作業	説明	参照先
対称鍵を生成します	ユーザーが指定したアルゴリズムで使用するランダム鍵を生成します。	294 ページの「dd コマンドを使用して対称鍵を生成する方法」
	ユーザーが指定した長さの鍵を生成します。任意で、ファイル、PKCS #11 キーストア、または NSS キーストアに鍵を格納します。	296 ページの「pktool コマンドを使用して対称鍵を生成する方法」
ファイルの整合性を保証するチェックサムを提供します	受信者のファイルのコピーが送信されたファイルと同一のものであることを検証します。	299 ページの「ファイルの要約を計算する方法」
メッセージ認証コード (MAC) でファイルを保護します	自分がメッセージの送信者であることを受信者に証明します。	301 ページの「ファイルの MAC を計算する方法」
ファイルを暗号化したあと、暗号化されたファイルを復号化します	ファイルを暗号化することによりファイルの内容を保護します。ファイルを復号化するための暗号化パラメータを指定します。	302 ページの「ファイルを暗号化および復号化する方法」

暗号化フレームワークによるファイルの保護 (手順)

この節では、対称鍵を生成する方法、ファイルの整合性のためにチェックサムを作成する方法、およびファイルが傍受されるのを防ぐ方法について説明します。この節のコマンドは、通常のユーザーが実行することができます。開発者は、これらのコマンドを使用するスクリプトを作成することができます。

▼ dd コマンドを使用して対称鍵を生成する方法

ファイルを暗号化し、ファイルの MAC を生成するには、鍵が必要です。鍵は、数のランダムプールから生成します。

乱数発生関数がすでにある場合は、それを使用してください。この関数がない場合は、Oracle Solaris の `/dev/urandom` デバイスを入力として `dd` コマンドを使用します。詳細は、dd(1M) のマニュアルページを参照してください。

1 アルゴリズムが必要とする鍵の長さを決定します。

a. 使用可能なアルゴリズムを一覧表示します。

```
% encrypt -l
Algorithm       Keysize:  Min   Max (bits)
----------------------------------------
aes                      128   128
arcfour                    8   128
des                       64    64
3des                     192   192

% mac -l
Algorithm       Keysize:  Min   Max (bits)
----------------------------------------
des_mac                   64    64
sha1_hmac                  8   512
md5_hmac                   8   512
sha256_hmac                8   512
sha384_hmac                8  1024
sha512_hmac                8  1024
```

b. **dd** コマンドに渡す鍵の長さをバイト単位で定義します。

最小鍵サイズと最大鍵サイズを 8 で割ります。最小鍵サイズと最大鍵サイズが異なるときは、中間の鍵サイズを使用することができます。たとえば、`sha1_hmac` 関数および `md5_hmac` 関数の場合、値 8、 16、 または 64 を `dd` コマンドに渡すことができます。

2 対称鍵を生成します。

```
% dd if=/dev/urandom of=keyfile bs=n count=n
```

`if=`*file* 入力ファイルです。ランダム鍵に対しては、`/dev/urandom` ファイルを使用します。

`of=`*keyfile* 生成された鍵を保持する出力ファイルです。

`bs=`*n* バイト単位の鍵サイズです。バイト単位の鍵サイズに対しては、ビット単位の鍵の長さを 8 で割ります。

`count=`*n* 入力ブロックの数です。*n* の数は 1 とします。

3 鍵を保護されたディレクトリに格納します。

そのユーザー以外は鍵のファイルを読み取ることができないように変更します。

```
% chmod 400 keyfile
```

例 14–1 AES アルゴリズムの鍵を作成する

次の例では、AES アルゴリズムの秘密鍵を作成します。この鍵は、あとで復号化するために格納も行います。AES メカニズムでは、 128 ビット鍵を使用します。この鍵は、`dd` コマンドでは 16 バイトです。

```
% ls -al ~/keyf
drwx------   2 jdoe  staff        512 May 3 11:32 ./
% dd if=/dev/urandom of=$HOME/keyf/05.07.aes16 bs=16  count=1
% chmod 400 ~/keyf/05.07.aes16
```

例 14–2 DES アルゴリズムの鍵を作成する

次の例では、DES アルゴリズムの秘密鍵を作成します。この鍵は、あとで復号化するために格納も行います。DES メカニズムでは、 64 ビット鍵を使用します。この鍵は、dd コマンドでは 8 バイトです。

```
% dd if=/dev/urandom of=$HOME/keyf/05.07.des8 bs=8  count=1
% chmod 400 ~/keyf/05.07.des8
```

例 14–3 3DES アルゴリズムの鍵を作成する

次の例では、3DES アルゴリズムの秘密鍵を作成します。この鍵は、あとで復号化するために格納も行います。3DES メカニズムでは、 192 ビット鍵を使用します。この鍵は、dd コマンドでは 24 バイトです。

```
% dd if=/dev/urandom of=$HOME/keyf/05.07.3des.24 bs=24 count=1
% chmod 400 ~/keyf/05.07.3des.24
```

例 14–4 MD5 アルゴリズムの鍵を作成する

次の例では、MD5 アルゴリズムの秘密鍵を作成します。この鍵は、あとで復号化するために格納も行います。この鍵は、dd コマンドでは 64 バイトです。

```
% dd if=/dev/urandom of=$HOME/keyf/05.07.mack64 bs=64 count=1
% chmod 400 ~/keyf/05.07.mack64
```

▼ pktool コマンドを使用して対称鍵を生成する方法

アプリケーションによっては、通信の暗号化および復号化に対称鍵が必要です。この手順では、対称鍵を作成して格納します。

- 乱数発生関数がある場合、この関数を使用して鍵の乱数を作成できます。この手順は乱数発生関数を使用しません。
- 代わりに、Oracle Solaris の /dev/urandom デバイスを入力として dd コマンドを使用します。dd コマンドは鍵を格納しません。手順については、294 ページの「dd コマンドを使用して対称鍵を生成する方法」を参照してください。

1 (省略可能) キーストアを使用する場合は、作成します。

- **PKCS #11** キーストアを作成して初期化する方法については、**326** ページの「**pktool setpin** コマンドを使ってパスフレーズを生成する方法」を参照してください。
- **NSS** データベースを作成して初期化する場合は、例 **15–5** を参照してください。

2 対称鍵として使用する乱数を生成します。
次のいずれかを実行します。

- 鍵を生成してファイルに格納します。
  鍵をファイルに格納する利点は、このファイルから鍵を抽出して、`/etc/inet/secret/ipseckeys` ファイルや IPsec など、アプリケーションの鍵ファイルで使用できることです。

```
% pktool genkey keystore=file outkey=key-fn \
[keytype=specific-symmetric-algorithm] [keylen=size-in-bits] \
[dir=directory] [print=n]
```

  `keystore`
  `file` の値は、鍵の格納場所のファイルタイプを指定します。

  `outkey=`*key-fn*
  `keystore=file` のときのファイル名です。

  `keytype=`*specific-symmetric-algorithm*
  特定のアルゴリズムとして、`aes`、 `arcfour`、 `des`、 または `3des` を指定します。

  `keylen=`*size-in-bits*
  鍵のビット長です。8 で割り切れる数にする必要があります。`des` または `3des` には指定しないでください。

  `dir=`*directory*
  key-fn へのディレクトリパスです。デフォルトでは、*directory* は現在のディレクトリです。

  `print=n`
  鍵を端末ウィンドウに印刷します。デフォルトでは、`print` の値は `n` です。

- 鍵を生成して **PKCS #11** キーストアに格納します。
  PKCS #11 キーストアの利点は、ラベルに基づいて鍵を取得できることです。この方法は、ファイルを暗号化および復号化する鍵の場合に便利です。この方法を使用するには、手順 1 を完了する必要があります。

```
% pktool genkey label=key-label \
[keytype=specific-symmetric-algorithm] [keylen=size-in-bits] \
[token=token] [sensitive=n] [extractable=y] [print=n]
```

`label=`*key-label*
鍵についてユーザーが指定したラベルです。ラベルに基づいてキーストアから鍵を取得できます。

`keytype=`*specific-symmetric-algorithm*
特定のアルゴリズムとして、`aes`、 `arcfour`、 `des`、 または `3des` を指定します。

`keylen=`*size-in-bits*
鍵のビット長です。8 で割り切れる数にする必要があります。`des` または `3des` には指定しないでください。

`token=`*token*
トークン名です。デフォルトでは、トークンは `Sun Software PKCS#11 softtoken` です。

`sensitive=n`
鍵の重要度を指定します。値が `y` の場合、鍵は `print=y` 引数を使用して印刷することはできません。デフォルトでは、`sensitive` の値は `n` です。

`extractable=y`
鍵がキーストアから抽出できることを指定します。鍵が抽出されないようにするには、`n` を指定します。

`print=n`
鍵を端末ウィンドウに印刷します。デフォルトでは、`print` の値は `n` です。

- 鍵を生成して **NSS** キーストアに格納します。

  この方法を使用するには、手順 1 を完了する必要があります。

```
% pktool keystore=nss genkey label=key-label \
[keytype=[keytype=specific-symmetric-algorithm] [keylen=size-in-bits] [token=token] \
[dir=directory-path] [prefix=database-prefix]
```

  `keystore`
  `nss` の値は、鍵の格納場所の NSS タイプを指定します。

  `label=`*key-label*
  鍵についてユーザーが指定したラベルです。ラベルに基づいてキーストアから鍵を取得できます。

  `keytype=`*specific-symmetric-algorithm*
  特定のアルゴリズムとして、`aes`、 `arcfour`、 `des`、 または `3des` を指定します。

  `keylen=`*size-in-bits*
  鍵のビット長です。8 で割り切れる数にする必要があります。`des` または `3des` には指定しないでください。

  `token=`*token*
  トークン名です。デフォルトでは、トークンは NSS 内部トークンです。

`dir=`*directory*
NSS データベースへのディレクトリパスです。デフォルトでは、*directory* は現在のディレクトリです。

`prefix=`*directory*
NSS データベースの接頭辞です。デフォルトは接頭辞なしです。

`print=n`
鍵を端末ウィンドウに印刷します。デフォルトでは、`print` の値は `n` です。

3 **(省略可能) 鍵が存在することを確認します。**

鍵を格納した場所に応じて、次のコマンドのいずれかを使用します。

- *key-fn* ファイル内の鍵を確認します。

```
% pktool list keystore=file objtype=key infile=key-fn
Found n keys.
Key #1 - keytype:location (keylen)
```

- **PKCS #11** または **NSS** キーストア内の鍵を確認します。

```
$ pktool list objtype=key
Enter PIN for keystore:
Found n keys.
Key #1 - keytype:location (keylen)
```

例 14–5 pktool コマンドを使用して DES 鍵を作成する

次の例では、DES アルゴリズムの秘密鍵を作成します。鍵は、あとで復号化するためにローカルファイルに格納されます。コマンドは、`400` のアクセス権でファイルを保護します。鍵が作成されると、`print=y` オプションにより、生成された鍵が端末ウィンドウに表示されます。

DES メカニズムでは、 64 ビット鍵を使用します。鍵ファイルを所有するユーザーは、`od` コマンドを使用して鍵を取得します。

```
% pktool genkey keystore=file outkey=64bit.file1 keytype=des print=y
        Key Value ="a3237b2c0a8ff9b3"
% od -x 64bit.file1
0000000 a323 7b2c 0a8f f9b3
```

▼ ファイルの要約を計算する方法

ファイルの要約を作成すると、要約の出力を比較することによって、ファイルが改ざんされていないことを確認することができます。要約によって元のファイルが変更されることはありません。

1 使用可能な要約アルゴリズムを列挙します。

```
% digest -l
md5
sha1
sha256
sha384
sha512
```

2 ファイルの要約を計算し、要約の一覧を保存します。

digest コマンドでアルゴリズムを指定します。

```
% digest -v -a algorithm input-file > digest-listing
```

-v 次の形式で出力を表示します。

algorithm (*input-file*) = *digest*

-a *algorithm* ファイルの要約の計算に使用するアルゴリズムです。手順 1 の出力のようにアルゴリズムを入力します。

input-file digest コマンドの入力ファイルです。

digest-listing digest コマンドの出力ファイルです。

例 14–6 MD5 メカニズムで要約を計算する

次の例では、digest コマンドにより、 MD5 メカニズムを使用して電子メールの添付ファイルの要約を計算します。

```
% digest -v -a md5 email.attach >>  $HOME/digest.emails.05.07
% cat  ~/digest.emails.05.07
md5 (email.attach) = 85c0a53d1a5cc71ea34d9ee7b1b28b01
```

-v オプションを使用しないと、要約は関連情報なしで保存されます。

```
% digest -a md5 email.attach >>  $HOME/digest.emails.05.07
% cat  ~/digest.emails.05.07
85c0a53d1a5cc71ea34d9ee7b1b28b01
```

例 14–7 SHA1 メカニズムで要約を計算する

次の例では、digest コマンドが SHA1 メカニズムを使用してディレクトリ一覧を提供します。結果はファイルに格納されます。

```
% digest -v -a sha1 docs/* > $HOME/digest.docs.legal.05.07
% more ~/digest.docs.legal.05.07
sha1 (docs/legal1) = 1df50e8ad219e34f0b911e097b7b588e31f9b435
sha1 (docs/legal2) = 68efa5a636291bde8f33e046eb33508c94842c38
sha1 (docs/legal3) = 085d991238d61bd0cfa2946c183be8e32cccf6c9
sha1 (docs/legal4) = f3085eae7e2c8d008816564fdf28027d10e1d983
```

▼ ファイルの MAC を計算する方法

メッセージ認証コード (MAC) は、ファイルの要約を計算し、秘密鍵を使用してさらに要約を保護します。MAC によって元のファイルが変更されることはありません。

1 使用可能なメカニズムを一覧表示します。

```
% mac -l
Algorithm       Keysize:  Min   Max
---------------------------------
des_mac                   64    64
sha1_hmac                  8   512
md5_hmac                   8   512
sha256_hmac                8   512
sha384_hmac                8  1024
sha512_hmac                8  1024
```

2 該当する長さの対称鍵を生成します。

2 つの選択肢があります。鍵が生成されるパスフレーズを指定することができます。または鍵を指定することができます。

- パスフレーズを指定する場合、指定したパスフレーズを格納するか覚えておく必要があります。パスフレーズをオンラインで格納する場合、そのパスフレーズファイルは自分だけが読み取ることができるようにします。
- 鍵を指定する場合、それはメカニズムの現在のサイズである必要があります。手順については、294 ページの「dd コマンドを使用して対称鍵を生成する方法」を参照してください。

3 ファイルの MAC を作成します。

mac コマンドで、鍵を指定して対称鍵アルゴリズムを使用します。

```
% mac -v -a algorithm [ -k keyfile ] input-file
```

`-v`	次の形式で出力を表示します。 *algorithm* (*input-file*) = *mac*
`-a` *algorithm*	MAC を計算するために使用するアルゴリズムです。`mac -l` コマンドの出力のようにアルゴリズムを入力します。
`-k` *keyfile*	アルゴリズムで指定された長さの鍵を含むファイルです。
input-file	MAC の入力ファイルです。

例 14–8 DES_MAC およびパスフレーズで MAC を計算する

次の例では、電子メールの添付ファイルを、DES_MAC メカニズムとパスフレーズから生成される鍵で認証します。MAC の一覧はファイルに保存されます。パスフレーズをファイルに格納する場合、そのファイルはユーザー以外は読み取ることができないようにします。

```
% mac -v -a des_mac email.attach
Enter passphrase:      <Type passphrase>
des_mac (email.attach) = dd27870a
% echo "des_mac (email.attach) = dd27870a" >> ~/desmac.daily.05.07
```

例 14–9 MD5_HMAC および鍵ファイルで MAC を計算する

次の例では、電子メールの添付ファイルを、MD5_HMAC メカニズムと秘密鍵で認証します。MAC の一覧はファイルに保存されます。

```
% mac -v -a md5_hmac -k $HOME/keyf/05.07.mack64 email.attach
md5_hmac (email.attach) = 02df6eb6c123ff25d78877eb1d55710c
% echo "md5_hmac (email.attach) = 02df6eb6c123ff25d78877eb1d55710c" \
>> ~/mac.daily.05.07
```

例 14–10 SHA1_HMAC および鍵ファイルで MAC を計算する

次の例では、ディレクトリの一覧を、SHA1_HMAC メカニズムと秘密鍵で認証します。結果はファイルに格納されます。

```
% mac -v -a sha1_hmac \
-k $HOME/keyf/05.07.mack64 docs/* > $HOME/mac.docs.legal.05.07
% more ~/mac.docs.legal.05.07
sha1_hmac (docs/legal1) = 9b31536d3b3c0c6b25d653418db8e765e17fe07a
sha1_hmac (docs/legal2) = 865af61a3002f8a457462a428cdb1a88c1b51ff5
sha1_hmac (docs/legal3) = 076c944cb2528536c9aebd3b9fbe367e07b61dc7
sha1_hmac (docs/legal4) = 7aede27602ef6e4454748cbd3821e0152e45beb4
```

▼ ファイルを暗号化および復号化する方法

ファイルを暗号化しても、元のファイルが削除されたり変更されたりすることはありません。出力ファイルが暗号化されます。

encrypt コマンドの一般的なエラーを解決するには、例のあとの節を参照してください。

1. 該当する長さの対称鍵を作成します。

   2 つの選択肢があります。鍵が生成されるパスフレーズを指定することができます。または鍵を指定することができます。

   - パスフレーズを指定する場合、指定したパスフレーズを格納するか覚えておく必要があります。パスフレーズをオンラインで格納する場合、そのパスフレーズファイルは自分だけが読み取ることができるようにします。

- 鍵を指定する場合、それはメカニズムの現在のサイズである必要があります。手順については、294 ページの「dd コマンドを使用して対称鍵を生成する方法」を参照してください。

2 ファイルを暗号化します。

encrypt コマンドで、鍵を指定して対称鍵アルゴリズムを使用します。

```
% encrypt -a algorithm  [ -k keyfile ] -i input-file -o output-file
```

-a *algorithm* ファイルを暗号化するために使用するアルゴリズムです。encrypt -l コマンドの出力のようにアルゴリズムを入力します。

-k *keyfile* アルゴリズムで指定された長さの鍵を含むファイルです。アルゴリズムごとの鍵の長さが、ビット単位で encrypt -l コマンドの出力に一覧表示されます。

-i *input-file* 暗号化する入力ファイルです。このファイルは、コマンドによって変更されることはありません。

-o *output-file* 入力ファイルと同じ暗号化形式の出力ファイルです。

例 14–11 AES およびパスフレーズで暗号化および復号化する

次の例では、ファイルを AES アルゴリズムで暗号化します。鍵はパスフレーズから生成されます。パスフレーズをファイルに格納する場合、そのファイルはユーザー以外は読み取ることができないようにします。

```
% encrypt -a aes -i ticket.to.ride -o ~/enc/e.ticket.to.ride
Enter passphrase:      <Type passphrase>
Re-enter passphrase:       Type passphrase again
```

入力ファイル ticket.to.ride は、元の形式で存在します。

出力ファイルを復号化する場合、ユーザーはファイルを暗号化したときと同じパスフレーズと暗号化メカニズムを使用します。

```
% decrypt -a aes -i ~/enc/e.ticket.to.ride -o ~/d.ticket.to.ride
Enter passphrase:      <Type passphrase>
```

例 14–12 AES および鍵ファイルで暗号化および復号化する

次の例では、ファイルを AES アルゴリズムで暗号化します。AES メカニズムでは、128 ビット(16 バイト) の鍵が使用されます。

```
% encrypt -a aes -k ~/keyf/05.07.aes16 \
-i ticket.to.ride -o ~/enc/e.ticket.to.ride
```

入力ファイル ticket.to.ride は、元の形式で存在します。

出力ファイルを復号化するとき、ユーザーはファイルを暗号化したときと同じ鍵と暗号化メカニズムを使用します。

```
% decrypt -a aes -k ~/keyf/05.07.aes16  \
-i ~/enc/e.ticket.to.ride -o ~/d.ticket.to.ride
```

例 14–13 ARCFOUR および鍵ファイルで暗号化および復号化する

次の例では、ファイルを ARCFOUR アルゴリズムで暗号化します。ARCFOUR アルゴリズムでは、8 ビット (1 バイト)、 64 ビット (8 バイト)、 または 128 ビット (16 バイト) の鍵が使用されます。

```
% encrypt -a arcfour -i personal.txt \
-k ~/keyf/05.07.rc4.8 -o ~/enc/e.personal.txt
```

出力ファイルを復号化するとき、ユーザーはファイルを暗号化したときと同じ鍵と暗号化メカニズムを使用します。

```
% decrypt -a arcfour -i ~/enc/e.personal.txt \
-k ~/keyf/05.07.rc4.8 -o ~/personal.txt
```

例 14–14 3DES および鍵ファイルで暗号化および復号化する

次の例では、ファイルを 3DES アルゴリズムで暗号化します。3DES アルゴリズムでは、192 ビット(24 バイト) の鍵が必要です。

```
% encrypt -a 3des -k ~/keyf/05.07.des24 \
-i ~/personal2.txt -o ~/enc/e.personal2.txt
```

出力ファイルを復号化するとき、ユーザーはファイルを暗号化したときと同じ鍵と暗号化メカニズムを使用します。

```
% decrypt -a 3des -k ~/keyf/05.07.des24 \
-i ~/enc/e.personal2.txt -o ~/personal2.txt
```

注意事項 次のメッセージでは、encryptコマンドに指定した鍵が、使用しているアルゴリズムで許可されないことが示されます。

- `encrypt: unable to create key for crypto operation: CKR_ATTRIBUTE_VALUE_INVALID`
- `encrypt: failed to initialize crypto operation: CKR_KEY_SIZE_RANGE`

アルゴリズムの条件を満たしていない鍵を渡した場合は、条件を満たす鍵を指定する必要があります。

- 1 つの方法として、パスフレーズを使用します。それによって、条件を満たす鍵が暗号化フレームワークで指定されます。

- 別の方法として、アルゴリズムで使用できる鍵サイズを渡します。たとえば、DES アルゴリズムでは 64 ビットの鍵が必要です。3DES アルゴリズムでは 192 ビットの鍵が必要です。

暗号化フレームワークの管理 (作業マップ)

次の作業マップでは、Oracle Solaris 暗号化フレームワークのソフトウェアプロバイダおよびハードウェアプロバイダの管理の手順を示します。

作業	説明	参照先
Oracle Solaris 暗号化フレームワークのプロバイダを一覧表示します	Oracle Solaris 暗号化フレームワークで使用可能なアルゴリズム、ライブラリ、およびハードウェアデバイスを一覧表示します。	306 ページの「使用可能なプロバイダを一覧表示する方法」
ソフトウェアプロバイダを追加します	PKCS #11 ライブラリまたはカーネルモジュールを Oracle Solaris 暗号化フレームワークに追加します。プロバイダは署名されている必要があります。	308 ページの「ソフトウェアプロバイダを追加する方法」
ユーザーレベルのメカニズムが使用されないようにします	ソフトウェアメカニズムの使用を解除します。ソフトウェアメカニズムは、再度有効にすることができます。	310 ページの「ユーザーレベルのメカニズムが使用されないようにする方法」
カーネルモジュールのメカニズムを一時的に無効にします	一時的にメカニズムの使用を解除します。通常はテストのために使用します。	311 ページの「カーネルソフトウェアプロバイダが使用されないようにする方法」
プロバイダをアンインストールします	カーネルソフトウェアプロバイダの使用を解除します。	例 14–22
使用可能なハードウェアプロバイダを一覧表示します	接続されているハードウェア、そのハードウェアに提供されているメカニズム、および使用可能になっているメカニズムを表示します。	314 ページの「ハードウェアプロバイダを一覧表示する方法」
ハードウェアプロバイダのメカニズムを無効にします	ハードウェアアクセラレータ上の選択したメカニズムが使用されないようにします。	315 ページの「ハードウェアプロバイダのメカニズムと機能を無効にする方法」
暗号化サービスを再起動または更新します	暗号化サービスを使用できるようにします。	316 ページの「すべての暗号化サービスを更新または再起動する方法」

暗号化フレームワークの管理 (手順)

この節では、Oracle Solaris 暗号化フレームワークでのソフトウェアプロバイダとハードウェアプロバイダの管理方法について説明します。必要に応じて、ソフトウェアプロバイダおよびハードウェアプロバイダの使用を解除することができます。たとえば、ソフトウェアプロバイダのアルゴリズムの実装を無効にすることができます。その後、別のソフトウェアプロバイダのアルゴリズムがシステムで使用されるようにすることができます。

▼ 使用可能なプロバイダを一覧表示する方法

Oracle Solaris 暗号化フレームワークには、数種類のコンシューマ用のアルゴリズムが用意されています。

- ユーザーレベルプロバイダは、`libpkcs11` ライブラリにリンクされているアプリケーションに PKCS #11 暗号化インタフェースを提供します
- カーネルソフトウェアプロバイダは、IPsec、 Kerberos、 およびその他の Oracle Solaris カーネルコンポーネント用のアルゴリズムを提供します
- カーネルハードウェアプロバイダは、`pkcs11_kernel` ライブラリによって、カーネルコンシューマおよびアプリケーションが使用可能なアルゴリズムを提供します

1 簡潔な形式でプロバイダを一覧表示します。

注 – プロバイダリストの内容と書式は、Oracle Solaris のリリースによって異なります。使用しているシステムでサポートされるプロバイダを表示するには、システムで `cryptoadm list` コマンドを実行します。

通常のユーザーは、ユーザーレベルのメカニズムのみを使用できます。

```
% cryptoadm list
user-level providers:
    /usr/lib/security/$ISA/pkcs11_kernel.so
    /usr/lib/security/$ISA/pkcs11_softtoken.so

kernel software providers:
    des
    aes
    blowfish
    arcfour
    sha1
    md5
    rsa

kernel hardware providers:
    ncp/0
```

2 **Oracle Solaris 暗号化フレームワークのプロバイダとそのメカニズムを一覧表示します。**

すべてのメカニズムが次の出力のように一覧表示されます。ただし、一覧表示されたメカニズムのいくつかは使用できない場合があります。管理者によって使用が許可されているメカニズムのみを一覧表示する場合は、例 14–16 を参照してください。

この出力は、表示用に整形されています。

```
% cryptoadm list -m
user-level providers:
=====================
/usr/lib/security/$ISA/pkcs11_kernel.so: CKM_MD5,CKM_MD5_HMAC,
CKM_MD5_HMAC_GENERAL,CKM_SHA_1,CKM_SHA_1_HMAC,CKM_SHA_1_HMAC_GENERAL,
...
/usr/lib/security/$ISA/pkcs11_softtoken.so:
CKM_DES_CBC,CKM_DES_CBC_PAD,CKM_DES_ECB,CKM_DES_KEY_GEN,
CKM_DES3_CBC,CKM_DES3_CBC_PAD,CKM_DES3_ECB,CKM_DES3_KEY_GEN,
CKM_AES_CBC,CKM_AES_CBC_PAD,CKM_AES_ECB,CKM_AES_KEY_GEN,
...
kernel software providers:
==========================
des: CKM_DES_ECB,CKM_DES_CBC,CKM_DES3_ECB,CKM_DES3_CBC
aes: CKM_AES_ECB,CKM_AES_CBC
blowfish: CKM_BF_ECB,CKM_BF_CBC
arcfour: CKM_RC4
sha1: CKM_SHA_1,CKM_SHA_1_HMAC,CKM_SHA_1_HMAC_GENERAL
md5: CKM_MD5,CKM_MD5_HMAC,CKM_MD5_HMAC_GENERAL
rsa: CKM_RSA_PKCS,CKM_RSA_X_509,CKM_MD5_RSA_PKCS,CKM_SHA1_RSA_PKCS
swrand: No mechanisms presented.

kernel hardware providers:
==========================
ncp/0: CKM_DSA,CKM_RSA_X_509,CKM_RSA_PKCS,CKM_RSA_PKCS_KEY_PAIR_GEN,
CKM_DH_PKCS_KEY_PAIR_GEN,CKM_DH_PKCS_DERIVE,CKM_EC_KEY_PAIR_GEN,
CKM_ECDH1_DERIVE,CKM_ECDSA
```

例 14–15 既存の暗号化メカニズムを検索する

次の例では、ユーザーレベルライブラリ pkcs11_softtoken が提供するすべてのメカニズムを一覧表示します。

```
% cryptoadm list -m provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
Mechanisms:
CKM_DES_CBC,CKM_DES_CBC_PAD,CKM_DES_ECB,CKM_DES_KEY_GEN,
CKM_DES3_CBC,CKM_DES3_CBC_PAD,CKM_DES3_ECB,CKM_DES3_KEY_GEN,
...
CKM_SSL3_KEY_AND_MAC_DERIVE,CKM_TLS_KEY_AND_MAC_DERIVE
```

例 14–16 使用可能な暗号化メカニズムを検索する

ポリシーによって、どのメカニズムが使用可能かが判断されます。ポリシーは、管理者によって設定されます。管理者は、特定のプロバイダのメカニズムを無効にす

ることができます。-p オプションを指定すると、管理者が設定したポリシーによって許可されているメカニズムのリストが表示されます。

```
% cryptoadm list -p
user-level providers:
=====================
/usr/lib/security/$ISA/pkcs11_kernel.so: all mechanisms are enabled.
random is enabled.
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled.
random is enabled.

kernel software providers:
==========================
des: all mechanisms are enabled.
aes: all mechanisms are enabled.
blowfish: all mechanisms are enabled.
arcfour: all mechanisms are enabled.
sha1: all mechanisms are enabled.
md5: all mechanisms are enabled.
rsa: all mechanisms are enabled.
swrand: random is enabled.

kernel hardware providers:
==========================
ncp/0: all mechanisms are enabled.
```

▼ ソフトウェアプロバイダを追加する方法

1. **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

   Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2. システムで使用可能なソフトウェアプロバイダを一覧表示します。

   ```
   % cryptoadm list
   user-level providers:
       /usr/lib/security/$ISA/pkcs11_kernel.so
       /usr/lib/security/$ISA/pkcs11_softtoken.so

   kernel software providers:
       des
       aes
       blowfish
       arcfour
       sha1
       md5
       rsa

   kernel hardware providers:
       ncp/0
   ```

3 **pkgadd** コマンドを使用して、プロバイダのパッケージを追加します。

```
# pkgadd -d /path/to/package  pkginst
```

パッケージには、Sun の証明書によって署名されたソフトウェアが含まれている必要があります。Sun の証明書を要求し、プロバイダに署名する場合は、『Oracle Solaris セキュリティーサービス開発ガイド』の付録 F「暗号化プロバイダのパッケージ化と署名」を参照してください。

パッケージには、一連のメカニズムを備えたほかのプロバイダが使用可能であることを暗号化フレームワークに通知するスクリプトが含まれます。パッケージの要件については、『Oracle Solaris セキュリティーサービス開発ガイド』の付録 F「暗号化プロバイダのパッケージ化と署名」を参照してください。

4 プロバイダを更新します。

ソフトウェアプロバイダを追加した場合や、ハードウェアおよびそのハードウェアに指定されているポリシーを追加した場合は、プロバイダを更新する必要があります。

```
# svcadm refresh svc:/system/cryptosvc
```

5 新しいプロバイダをリストに追加します。

この場合、新しいカーネルソフトウェアプロバイダがインストールされています。

```
# cryptoadm list
...
kernel software providers:
    des
    aes
    blowfish
    arcfour
    sha1
    md5
    rsa
    swrand
    ecc       <-- added provider
...
```

例 14–17 ユーザーレベルソフトウェアプロバイダを追加する

次の例では、署名された PKCS #11 ライブラリをインストールします。

```
# pkgadd -d /cdrom/cdrom0/SolarisNew
    Answer the prompts
# svcadm refresh system/cryptosvc
# cryptoadm list
user-level providers:
=========================
    /usr/lib/security/$ISA/pkcs11_kernel.so
    /usr/lib/security/$ISA/pkcs11_softtoken.so
    /opt/SUNWconn/lib/$ISA/libpkcs11.so.1       <-- added provider
```

暗号化フレームワークによってライブラリをテストする開発者は、ライブラリを手動でインストールすることができます。

```
# cryptoadm install provider=/opt/SUNWconn/lib/\$ISA/libpkcs11.so.1
```

プロバイダの署名については、290 ページの「Sun 以外のソフトウェアのためのバイナリ署名」を参照してください。

▼ ユーザーレベルのメカニズムが使用されないようにする方法

ライブラリプロバイダの暗号化メカニズムに使用すべきでないものが存在する場合、選択したメカニズムを削除することができます。この手順では、一例として、pkcs11_softtoken ライブラリの DES メカニズムを使用します。

1. スーパーユーザーになるか、**Crypto Management** 権利プロファイルを含む役割を引き受けます。

   Crypto Management 権利プロファイルを含む役割を作成し、その役割をユーザーに割り当てる場合は、例 9–7 を参照してください。

2. 特定のユーザーレベルソフトウェアプロバイダによって提供されるメカニズムを一覧表示します。

   ```
   % cryptoadm list -m provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
   /usr/lib/security/$ISA/pkcs11_softtoken.so:
   CKM_DES_CBC,CKM_DES_CBC_PAD,CKM_DES_ECB,CKM_DES_KEY_GEN,
   CKM_DES3_CBC,CKM_DES3_CBC_PAD,CKM_DES3_ECB,CKM_DES3_KEY_GEN,
   CKM_AES_CBC,CKM_AES_CBC_PAD,CKM_AES_ECB,CKM_AES_KEY_GEN,
   ...
   ```

3. 使用可能なメカニズムを一覧表示します。

   ```
   $ cryptoadm list -p
   user-level providers:
   =====================
   ...
   /usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled.
   random is enabled.
   ...
   ```

4. 使用すべきでないメカニズムを無効にします。

   ```
   $ cryptoadm disable provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so \
   > mechanism=CKM_DES_CBC,CKM_DES_CBC_PAD,CKM_DES_ECB
   ```

5. 使用可能なメカニズムを一覧表示します。

   ```
   $ cryptoadm list -p provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
   /usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled,
   except CKM_DES_ECB,CKM_DES_CBC_PAD,CKM_DES_CBC. random is enabled.
   ```

例 14–18 ユーザーレベルソフトウェアプロバイダのメカニズムを有効にする

次の例では、無効になっている DES メカニズムを再び使用可能にします。

```
$ cryptoadm list -m provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so:
CKM_DES_CBC,CKM_DES_CBC_PAD,CKM_DES_ECB,CKM_DES_KEY_GEN,
CKM_DES3_CBC,CKM_DES3_CBC_PAD,CKM_DES3_ECB,CKM_DES3_KEY_GEN,
...
$ cryptoadm list -p provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled,
except CKM_DES_ECB,CKM_DES_CBC_PAD,CKM_DES_CBC. random is enabled.
$ cryptoadm enable provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so \
> mechanism=CKM_DES_ECB
$ cryptoadm list -p provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled,
except CKM_DES_CBC_PAD,CKM_DES_CBC. random is enabled.
```

例 14–19 ユーザーレベルソフトウェアプロバイダのメカニズムをすべて有効にする

次の例では、ユーザーレベルライブラリのメカニズムをすべて有効にします。

```
$ cryptoadm enable provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so all
$ cryptoadm list -p provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled.
random is enabled.
```

例 14–20 ユーザーレベルソフトウェアプロバイダの使用を永続的に削除する

次の例では、`libpkcs11.so.1` ライブラリを削除します。

```
$ cryptoadm uninstall provider=/opt/SUNWconn/lib/\$ISA/libpkcs11.so.1
$ cryptoadm list
user-level providers:
    /usr/lib/security/$ISA/pkcs11_kernel.so
    /usr/lib/security/$ISA/pkcs11_softtoken.so

kernel software providers:
...
```

▼ カーネルソフトウェアプロバイダが使用されないようにする方法

暗号化フレームワークに AES などのプロバイダの複数のモードが用意されている場合、遅いメカニズムや破壊されたメカニズムの使用を解除する場合があります。この手順では、一例として、AES アルゴリズムを使用します。

1 スーパーユーザーになるか、**Crypto Management** 権利プロファイルを含む役割を引き受けます。

Crypto Management 権利プロファイルを含む役割を作成し、その役割をユーザーに割り当てる場合は、例 9–7 を参照してください。

2 特定のカーネルソフトウェアプロバイダによって提供されるメカニズムを一覧表示します。

```
$ cryptoadm list -m provider=aes
aes: CKM_AES_ECB,CKM_AES_CBC
```

3 使用可能なメカニズムを一覧表示します。

```
$ cryptoadm list -p provider=aes
aes: all mechanisms are enabled.
```

4 使用すべきでないメカニズムを無効にします。

```
$ cryptoadm disable provider=aes mechanism=CKM_AES_ECB
```

5 使用可能なメカニズムを一覧表示します。

```
$ cryptoadm list -p provider=aes
aes: all mechanisms are enabled, except CKM_AES_ECB.
```

例 14–21 カーネルソフトウェアプロバイダのメカニズムを有効にする

次の例では、無効になっている AES メカニズムを再び使用可能にします。

```
cryptoadm list -m provider=aes
aes: CKM_AES_ECB,CKM_AES_CBC
$ cryptoadm list -p provider=aes
aes: all mechanisms are enabled, except CKM_AES_ECB.
$ cryptoadm enable provider=aes mechanism=CKM_AES_ECB
$ cryptoadm list -p provider=aes
aes: all mechanisms are enabled.
```

例 14–22 カーネルソフトウェアプロバイダの使用を一時的に削除する

次の例では、AES プロバイダの使用を一時的に削除します。`unload` サブコマンドは、プロバイダのアンインストール中にプロバイダが自動的に読み込まれないようにするために使用します。たとえば、`unload` サブコマンドは、プロバイダに影響を与えるパッチをインストールするときに使用します。

```
$ cryptoadm unload provider=aes

$ cryptoadm list
...
kernel software providers:
    des
    aes (inactive)
    blowfish
```

```
    arcfour
    sha1
    md5
    rsa
    swrand
```

AES プロバイダは、暗号化フレームワークが更新されるまでは使用できません。

```
$ svcadm refresh system/cryptosvc

$ cryptoadm list
...
kernel software providers:
    des
    aes
    blowfish
    arcfour
    sha1
    md5
    rsa
    swrand
```

カーネルコンシューマがカーネルソフトウェアプロバイダを使用している場合は、ソフトウェアは読み込み解除されません。エラーメッセージが表示され、プロバイダを使用し続けることができます。

例 14–23 ソフトウェアプロバイダの使用を永続的に解除する

次の例では、AES プロバイダの使用を解除します。いったん削除すると、AES プロバイダはカーネルソフトウェアプロバイダのポリシー一覧に表示されません。

```
$ cryptoadm uninstall provider=aes

$ cryptoadm list
...
kernel software providers:
    des
    blowfish
    arcfour
    sha1
    md5
    rsa
    swrand
```

カーネルコンシューマがカーネルソフトウェアプロバイダを使用している場合は、エラーメッセージが表示され、プロバイダを使用し続けることができます。

例 14–24 削除されたカーネルソフトウェアプロバイダを再インストールする

次の例では、AES カーネルソフトウェアプロバイダを再インストールします。

```
$ cryptoadm install provider=aes mechanism=CKM_AES_ECB,CKM_AES_CBC
```

```
$ cryptoadm list
...
kernel software providers:
    des
    aes
    blowfish
    arcfour
    sha1
    md5
    rsa
    swrand
```

▼ ハードウェアプロバイダを一覧表示する方法

ハードウェアプロバイダは、自動的に配置され読み込まれます。詳細は、`driver.conf(4)` のマニュアルページを参照してください。

始める前に　Oracle Solaris 暗号化フレームワーク内での使用が想定されているハードウェアがある場合、そのハードウェアはカーネルの SPI に登録されます。暗号化フレームワークでは、ハードウェアドライバが署名されていることが確認されます。特に、ドライバのオブジェクトファイルが Sun が発行する証明書付きで署名されていることが確認されます。

たとえば、Sun Crypto Accelerator 6000 ボード (`mca`)、 UltraSPARC T1 および T2 プロセッサの暗号化アクセラレータ用 ncp ドライバ (`ncp`)、 UltraSPARC T2 プロセッサ用 n2cp ドライバ (`n2cp`) は、ハードウェアのメカニズムをフレームワークに接続します。

プロバイダの署名については、290 ページの「Sun 以外のソフトウェアのためのバイナリ署名」を参照してください。

1. システムで使用可能なハードウェアプロバイダを一覧表示します。

```
% cryptoadm list
...
kernel hardware providers:
   ncp/0
```

2. チップまたはボードで提供されるメカニズムを一覧表示します。

```
% cryptoadm list -m provider=ncp/0
ncp/0: CKM_DSA,CKM_RSA_X_509,CKM_RSA_PKCS,CKM_RSA_PKCS_KEY_PAIR_GEN,
CKM_DH_PKCS_KEY_PAIR_GEN,CKM_DH_PKCS_DERIVE,CKM_EC_KEY_PAIR_GEN,
CKM_ECDH1_DERIVE,CKM_ECDSA
```

3. チップまたはボードで使用可能なメカニズムを一覧表示します。

```
% cryptoadm list -p provider=ncp/0
ncp/0: all mechanisms are enabled.
```

▼ ハードウェアプロバイダのメカニズムと機能を無効にする方法

ハードウェアプロバイダのメカニズムや乱数機能を選択して無効にすることができます。それらを再び有効にする場合は、例 14–25 を参照してください。この例のハードウェア Sun Crypto Accelerator 1000 ボードは、乱数発生関数を提供します。

1 スーパーユーザーになるか、**Crypto Management** 権利プロファイルを含む役割を引き受けます。

Crypto Management 権利プロファイルを含む役割を作成し、その役割をユーザーに割り当てる場合は、例 9–7 を参照してください。

2 無効にするメカニズムまたは機能を選択します。

ハードウェアプロバイダを一覧表示します。

```
# cryptoadm list
...
Kernel hardware providers:
    dca/0
```

- 選択したメカニズムを無効にします。

```
# cryptoadm list -m provider=dca/0
dca/0: CKM_RSA_PKCS, CKM_RSA_X_509, CKM_DSA, CKM_DES_CBC, CKM_DES3_CBC
random is enabled.
# cryptoadm disable provider=dca/0 mechanism=CKM_DES_CBC,CKM_DES3_CBC
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled except CKM_DES_CBC,CKM_DES3_CBC.
random is enabled.
```

- 乱数発生関数を無効にします。

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is enabled.
# cryptoadm disable provider=dca/0 random
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is disabled.
```

- すべてのメカニズムを無効にします。乱数発生関数は無効にしません。

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is enabled.
# cryptoadm disable provider=dca/0 mechanism=all
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are disabled. random is enabled.
```

- ハードウェアのすべての機能とメカニズムを無効にします。

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is enabled.
# cryptoadm disable provider=dca/0 all
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are disabled. random is disabled.
```

例 14–25 ハードウェアプロバイダのメカニズムと機能を有効にする

次の例では、一部のハードウェアの無効になっているメカニズムを選択して有効にします。

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled except CKM_DES_ECB,CKM_DES3_ECB

.
random is enabled.
# cryptoadm enable provider=dca/0 mechanism=CKM_DES3_ECB
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled except CKM_DES_ECB.
random is enabled.
```

次の例では、乱数発生関数のみを有効にします。

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_MD5,CKM_MD5_HMAC,....
random is disabled.
# cryptoadm enable provider=dca/0 random
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_MD5,CKM_MD5_HMAC,....
random is enabled.
```

次の例では、メカニズムのみを有効にします。乱数発生関数は無効にしておきます。

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_MD5,CKM_MD5_HMAC,....
random is disabled.
# cryptoadm enable provider=dca/0 mechanism=all
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is disabled.
```

次の例では、ボードのすべての機能とメカニズムを有効にします。

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_DES_ECB,CKM_DES3_ECB.
random is disabled.
# cryptoadm enable provider=dca/0 all
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is enabled.
```

▼ すべての暗号化サービスを更新または再起動する方法

デフォルトでは、Oracle Solaris 暗号化フレームワークは有効になっています。何らかの理由で kcfd デーモンが失敗したときには、サービス管理機能を使用して暗号化サービスを再起動することができます。詳細は、smf(5) および svcadm(1M) のマ

ニュアルページを参照してください。暗号化サービスの再起動がゾーンに与える影響については、291 ページの「暗号化サービスとゾーン」を参照してください。

1 暗号化サービスの状態を確認します。

```
% svcs cryptosvc
 STATE          STIME    FMRI
offline         Dec_09   svc:/system/cryptosvc:default
```

2 スーパーユーザーになるか、同等の役割を引き受けて、暗号化サービスを有効にします。

役割には、認証と特権コマンドが含まれます。役割の詳細は、210 ページの「RBAC の構成 (作業マップ)」を参照してください。

```
# svcadm enable svc:/system/cryptosvc
```

例 14–26 暗号化サービスを更新する

次の例では、暗号化サービスを大域ゾーンで更新します。その結果、すべての非大域ゾーンのカーネルレベルの暗号化ポリシーも更新されます。

```
# svcadm refresh system/cryptosvc
```

◆ ◆ ◆ 第 15 章

15 Oracle Solaris 鍵管理フレームワーク

Solaris 10 8/07 リリースから、鍵管理フレームワーク (KMF) を通じて、公開鍵オブジェクトを管理するためのツールとプログラミングインタフェースが提供されています。公開鍵オブジェクトには、X.509 証明書や公開鍵/非公開鍵ペアが含まれます。これらのオブジェクトの格納形式としては、さまざまなものが使えます。また、KMF では、アプリケーションによる X.509 証明書の使用方法を定義したポリシーを管理するためのツールも提供されます。

公開鍵技術の管理

鍵管理フレームワーク (KMF) は、公開鍵技術 (PKI) を管理するための統一されたアプローチを提供します。Oracle Solaris には、PKI 技術を利用する異なるアプリケーションがいくつか含まれています。各アプリケーションはそれぞれ独自のプログラミングインタフェース、鍵格納メカニズム、および管理ユーティリティーを提供します。アプリケーションがあるポリシー適用メカニズムを提供する場合、そのメカニズムはそのアプリケーションにしか適用されません。KMF では、アプリケーションは統一された管理ツール群、単一のプログラミングインタフェース群、および単一のポリシー適用メカニズムを使用します。これらのインタフェースを採用したすべてのアプリケーションの PKI ニーズは、これらの機能によって管理されます。

KMF では、次のインタフェースによって公開鍵技術の管理が統一されます。

- **pktool** コマンド – このコマンドは各種キーストア内の、証明書などの PKI オブジェクトを管理します。
- **kmfcfg** コマンド – このコマンドは PKI ポリシーデータベースを管理します。

  PKI ポリシー決定には、ある操作の検証方法などの操作が含まれます。また、PKI ポリシーは証明書の適用範囲を制限することもできます。たとえば、ある証明書が特定の目的にしか使用できないことを主張する PKI ポリシーを定義できます。そのようなポリシーは、その証明書がほかの要求のために使用されるのを禁止します。
- **KMF** ライブラリ – このライブラリには、ベースとなるキーストアメカニズムを抽象化するプログラミングインタフェースが含まれています。

  アプリケーションは特定のキーストアメカニズムを選択する必要がなく、あるメカニズムから別のメカニズムへ移行できます。サポートされているキーストアは、PKCS #11、 NSS、 および OpenSSL です。このライブラリに含まれるプラグイン可能フレームワークを使えば、新しいキーストアメカニズムを追加できます。したがって、アプリケーションがある新しいメカニズムを使用する場合、アプリケーションを若干変更するだけで、その新しいキーストアを使用できるようになります。

鍵管理フレームワークのユーティリティー

KMF は、鍵の格納を管理するための手段を提供するとともに、それらの鍵の使用に関する包括的なポリシーを提供します。KMF は、次の 3 つの公開鍵技術のポリシー、鍵、および証明書を管理します。

- PKCS #11 プロバイダ、つまり Oracle Solaris 暗号化フレームワークからのトークン
- NSS、 つまりネットワークセキュリティーサービス
- OpenSSL、 ファイルベースのキーストア

`kmfcfg` ツールを使えば、KMF ポリシーエントリの作成、変更、または削除を行えます。KMF は、`pktool` コマンド経由でキーストアを管理します。詳細については、`kmfcfg(1)` と `pktool(1)` のマニュアルページ、および次の各節を参照してください。

KMF のポリシー管理

KMF のポリシーはデータベース内に格納されます。このポリシーデータベースは、KMF プログラミングインタフェースを使用するすべてのアプリケーションによって内部的にアクセスされます。このデータベースを使えば、KMF ライブラリによって管理される鍵や証明書の使用に、制約を設けることができます。アプリケーションは、証明書の検証を試みる際に、ポリシーデータベースをチェックします。`kmfcfg` コマンドはポリシーデータベースを変更します。

KMF のキーストア管理

KMF は、PKCS #11 トークン、NSS、 OpenSSL の 3 つの公開鍵技術のキーストアを管理します。`pktool` コマンドを使えば、これらすべての技術について次のことが行えます。

- 自己署名付き証明書を生成します。
- 証明書要求を生成します。
- キーストアにオブジェクトをインポートします。
- キーストア内のオブジェクトを一覧表示します。
- キーストアからオブジェクトを削除します。
- CRL をダウンロードします。

PKCS #11 および NSS 技術の場合には、`pktool` コマンドでパスフレーズを生成して PIN を設定することもできます。

- キーストアのパスフレーズを生成します。
- キーストア内のあるオブジェクトのパスフレーズを生成します。

`pktool` ユーティリティーの使用例については、`pktool(1)` のマニュアルページおよび321 ページの「鍵管理フレームワークの使用 (作業マップ)」を参照してください。

鍵管理フレームワークの使用 (作業マップ)

鍵管理フレームワーク (KMF) を使うと公開鍵技術を集中管理できます。

作業	説明	参照先
証明書を作成します。	PKCS #11、 NSS、 SSL で使用される証明書を作成します。	322 ページの「`pktool gencert` コマンドを使って証明書を作成する方法」
証明書をエクスポートします。	証明書とそれをサポートする鍵を含むファイルを作成します。このファイルはパスワードで保護できます。	325 ページの「証明書と非公開鍵を PKCS #12 形式でエクスポートする方法」

作業	説明	参照先
証明書をインポートします。	別のシステムから取得した証明書をインポートします。	323 ページの「証明書をキーストアにインポートする方法」
	別のシステムから取得した PKCS #12 形式の証明書をインポートします。	例 15–2
パスフレーズを生成します。	PKCS #11 キーストアまたは NSS キーストアにアクセスするためのパスフレーズを生成します。	326 ページの「pktool setpin コマンドを使ってパスフレーズを生成する方法」

鍵管理フレームワークの使用 (手順)

この節では、パスワード、パスフレーズ、ファイル、キーストア、証明書、CRL などの公開鍵オブジェクトを、pktool コマンドを使って管理する方法について説明します。

▼ pktool gencert コマンドを使って証明書を作成する方法

この手順では、自己署名付き証明書を作成し、その証明書を PKCS #11 キーストアに格納します。また、この処理の一部として、RSA 公開鍵/非公開鍵ペアも作成されます。非公開鍵は、証明書とともにキーストアに格納されます。

1 自己署名付き証明書を生成します。

```
% pktool gencert [keystore=keystore] label=label-name \
subject=subject-DN serial=hex-serial-number
```

keystore=*keystore*	公開鍵オブジェクトのタイプを指定することでキーストアを指定します。指定可能な値は nss、 pkcs11、 ssl のいずれかです。このキーワードは省略可能です。
label=*label-name*	発行者が証明書に与える一意の名前を指定します。
subject=*subject-DN*	証明書の識別名を指定します。
serial=*hex-serial-number*	16 進形式のシリアル番号を指定します。証明書の発行者が、0x0102030405 などの番号を選択します。

2 キーストアの内容を確認します。

```
% pktool list
Found number certificates.
1. (X.509 certificate)
      Label:  label-name
      ID:  Fingerprint that binds certificate to private key
```

```
      Subject: subject-DN
      Issuer:  distinguished-name
      Serial:  hex-serial-number
n. ...
```

このコマンドは、キーストア内のすべての証明書を一覧表示します。次の例では、キーストアには証明書が 1 つしか含まれていません。

例 15–1 pktool を使って自己署名付き証明書を作成する

次の例では、My Company のあるユーザーが自己署名付き証明書を作成し、その証明書を PKCS #11 オブジェクト用のキーストアに格納しています。このキーストアは最初は空になっています。キーストアが初期化されていない場合、ソフトトークンの PIN は changeme になります。

```
% pktool gencert keystore=pkcs11 label="My Cert" \
subject="C=US, O=My Company, OU=Security Engineering Group, CN=MyCA" \
serial=0x000000001
Enter pin for Sun Software PKCS#11 softtoken:     Type PIN for token

% pktool list
Found 1 certificates.
1. (X.509 certificate)
      Label: My Cert
      ID: 12:82:17:5f:80:78:eb:44:8b:98:e3:3c:11:c0:32:5e:b6:4c:ea:eb
      Subject: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
      Issuer: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
      Serial: 0x01
```

▼ 証明書をキーストアにインポートする方法

この手順では、PEM または生の DER を使ってエンコードされた PKI 情報を含むファイルを、キーストアにインポートする方法を説明します。エクスポートの手順については、例 15–4 を参照してください。

1. 証明書をインポートします。

   ```
   % pktool import keystore=keystore infile=infile-name label=label-name
   ```

2. 非公開 **PKI** オブジェクトをインポートする場合、プロンプトが表示されたときにパスワードを入力します。

   a. プロンプトで、ファイルのパスワードを入力します。

      PKCS #12 形式のエクスポートファイルなど、非公開の PKI 情報をインポートする場合、そのファイルはパスワードを必要とします。インポートするファイルの作成者が PKCS #12 パスワードを教えてくれます。

      ```
      Enter password to use for accessing the PKCS12 file:     Type PKCS #12 password
      ```

b. プロンプトで、キーストアのパスワードを入力します。

```
Enter pin for Sun Software PKCS#11 softtoken:      Type PIN for token
```

3 キーストアの内容を確認します。

```
% pktool list
Found number certificates.
1. (X.509 certificate)
      Label:  label-name
      ID: Fingerprint that binds certificate to private key
      Subject: subject-DN
      Issuer:  distinguished-name
      Serial:  hex-serial-number
2. ...
```

例 15–2 PKCS #12 ファイルをキーストアにインポートする

次の例では、サードパーティーから取得した PKCS #12 ファイルをインポートしています。pktool import コマンドは、gracedata.p12 ファイルから非公開鍵と証明書を取り出し、それらを指定されたキーストア内に格納します。

```
% pktool import keystore=pkcs11 infile=gracedata.p12 label=GraceCert
Enter password to use for accessing the PKCS12 file:     Type PKCS #12 password
Enter pin for Sun Software PKCS#11 softtoken:      Type PIN for token
Found 1 certificate(s) and 1 key(s) in gracedata.p12
% pktool list
Found 1 certificates.
1. (X.509 certificate)
        Label: GraceCert
        ID: 12:82:17:5f:80:78:eb:44:8b:98:e3:3c:11:c0:32:5e:b6:4c:ea:eb
        Subject: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
        Issuer: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
        Serial: 0x01
```

例 15–3 X.509 証明書をキーストアにインポートする

次の例では、PEM 形式の X.509 証明書を指定されたキーストアにインポートしています。この公開証明書はパスワードで保護されていません。ユーザーの公開キーストアもパスワードで保護されていません。

```
% pktool import keystore=pkcs11 infile=somecert.pem label="TheirCompany Root Cert"
% pktool list
Found 1 certificates.
1. (X.509 certificate)
        Label: TheirCompany Root Cert
        ID: 21:ae:83:98:24:d1:1f:cb:65:5b:48:75:7d:02:47:cf:98:1f:ec:a0
        Subject: C=US, O=TheirCompany, OU=Security, CN=TheirCompany Root CA
        Issuer: C=US, O=TheirCompany, OU=Security, CN=TheirCompany Root CA
        Serial: 0x01
```

▼ 証明書と非公開鍵を PKCS #12 形式でエクスポートする方法

PKCS #12 形式のファイルを作成し、非公開鍵とそれに関連付けられた X.509 証明書をほかのシステムにエクスポートすることができます。このファイルへのアクセスはパスワードで保護されます。

1 エクスポートする証明書を見つけます。

```
% pktool list
Found number certificates.
1. (X.509 certificate)
      Label:  label-name
      ID:  Fingerprint that binds certificate to private key
      Subject:  subject-DN
      Issuer:  distinguished-name
      Serial:  hex-serial-number
2. ...
```

2 鍵と証明書をエクスポートします。

`pktool list` コマンドから得られたキーストアとラベルを使用します。エクスポートファイルのファイル名を指定します。名前に空白が含まれている場合は、名前を二重引用符で囲みます。

```
% pktool export keystore=keystore outfile=outfile-name label=label-name
```

3 エクスポートファイルをパスワードで保護します。

プロンプトで、キーストアの現在のパスワードを入力します。ここで、エクスポートファイルのパスワードを作成します。ファイルの受信者は、インポート時にこのパスワードを入力する必要があります。

```
Enter pin for Sun Software PKCS#11 softtoken:      Type PIN for token
Enter password to use for accessing the PKCS12 file:     Create PKCS #12 password
```

ヒント – パスワードはエクスポートファイルとは別に送ってください。パスワードを伝える最良の方法は、電話上など、通常の通信手段以外の手段を使用する方法です。

例 15–4 証明書と非公開鍵を PKCS #12 形式でエクスポートする

次の例では、あるユーザーが、非公開鍵とそれに関連付けられた X.509 証明書を、標準 PKCS #12 ファイル内にエクスポートしています。このファイルは、ほかのキーストアにインポートできます。PKCS #11 パスワードはソースのキーストアを保護します。PKCS #12 パスワードは、PKCS #12 ファイル内の非公開データを保護するために使用されます。このパスワードはファイルのインポート時に必要となります。

```
% pktool list
Found 1 certificates.
1. (X.509 certificate)
      Label: My Cert
      ID: 12:82:17:5f:80:78:eb:44:8b:98:e3:3c:11:c0:32:5e:b6:4c:ea:eb
      Subject: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
      Issuer: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
      Serial: 0x01

% pktool export keystore=pkcs11 outfile=mydata.p12 label="My Cert"
Enter pin for Sun Software PKCS#11 softtoken:      Type PIN for token
Enter password to use for accessing the PKCS12 file:      Create PKCS #12 password
```

続いて、このユーザーは受信者に電話をかけ、PKCS #12 パスワードを伝えます。

▼ pktool setpin コマンドを使ってパスフレーズを生成する方法

キーストア内のあるオブジェクトに対して、あるいはキーストアそのものに対して、パスフレーズを生成することができます。このパスフレーズは、オブジェクトやキーストアにアクセスする際に必要となります。キーストア内のオブジェクトに対するパスフレーズの生成例については、例 15–4 を参照してください。

1 キーストアにアクセスするためのパスフレーズを生成します。

```
% pktool setpin keystore=nss|pkcs11 dir=directory
```

2 プロンプトに答えます。

キーストアにまだパスワードが設定されていない場合には、Return キーを押してパスワードを作成します。

```
Enter current token passphrase:      Press the Return key
Create new passphrase:      Type the passphrase that you want to use
Re-enter new passphrase:      Retype the passphrase
Passphrase changed.
```

これでキーストアがパスフレーズで保護されます。パスフレーズを忘れると、キーストア内のオブジェクトにアクセスできなくなります。

例 15–5 キーストアをパスフレーズで保護する

次の例は、NSS データベースのパスフレーズを設定する方法を示したものです。パスフレーズがまだ作成されていないため、最初のプロンプトで Return キーを押します。

```
% pktool setpin keystore=nss dir=/var/nss
Enter current token passphrase:      Press the Return key
```

```
Create new passphrase:    has8n0NdaH
Re-enter new passphrase:  has8n0NdaH
Passphrase changed.
```

パート V

認証サービスと安全な通信

このパートでは、ネットワークに接続されていないシステム、または 2 つのシステム間で設定される認証サービスについて説明します。

- 第 16 章「認証サービスの使用 (手順)」
- 第 17 章「PAM の使用」
- 第 18 章「SASL の使用」
- 第 19 章「Oracle Solaris Secure Shell の使用 (手順)」
- 第 20 章「Oracle Solaris Secure Shell (参照)」

認証されたユーザーとシステムのネットワークを設定するには、パート VI「Kerberos サービス」を参照してください。

◆ ◆ ◆ 第 16 章

16 認証サービスの使用 (手順)

この章では、Secure RPC を使用して NFS マウントでホストとユーザーを認証する方法について説明します。この章で説明する内容は次のとおりです。

Secure RPC の概要

Secure RPC (遠隔手続き呼び出し) は、認証メカニズムによって遠隔手続きを保護します。Diffie-Hellman 認証メカニズムは、サービスを要求するホストとユーザーを認証します。この認証メカニズムはデータ暗号化規格 (Data Encryption Standard、 DES) 暗号化を使用します。Secure RPC を使用するアプリケーションには、NFS、 およびネームサービスの NIS と NIS+ があります。

NFS サービスと Secure RPC

NFS を使用すると、複数のホストがネットワーク上でファイルを共有できます。NFS サービスでは、サーバーは、複数のクライアントから利用できるデータとリソースを保持します。クライアントは、サーバーがクライアントと共有するファイルシステムにアクセスできます。クライアントシステムにログインしたユーザーは、ファイルシステムをサーバーからマウントすることによって、そのファイルシステムにアクセスできます。このとき、クライアントシステム上のユーザーには、ファイルはクライアントのローカルファイルシステム上にあるように見えます。NFS のもっとも一般的な使用形態の 1 つは、システムを各オフィスにインストールして、すべてのユーザーファイルを 1 箇所で集中管理することです。`mount` コマンドの `-nosuid` オプションなどのいくつかの NFS 機能を使用すると、権限を持たないユーザーがデバイスやファイルシステムにアクセスすることを禁止できます。

NFS サービスでは Secure RPC を使用して、要求を出したユーザーをネットワーク上で認証します。このプロセスは、*Secure NFS* と呼ばれます。Diffie-Hellman 認証メカニズム AUTH_DH は、DES 暗号化を使用し、承認されたアクセスを保証します。AUTH_DH メカニズムは、AUTH_DES とも呼ばれます。詳細については、次を参照してください。

- Secure NFS の設定と管理については、『Solaris のシステム管理 (ネットワークサービス)』の「Secure NFS システムの管理」を参照してください。
- NIS+ テーブルの設定と cred テーブルへの名前の入力については、『Solaris のシステム管理 (ネーミングとディレクトリサービス : NIS+ 編)』を参照してください。
- RPC 認証トランザクションの概要については、333 ページの「Diffie-Hellman 認証の実装」を参照してください。

Secure NFS での DES 暗号化

データ暗号化規格 (Data Encryption Standard、 DES) 暗号化機能は 56 ビットの鍵を使用して、データを暗号化します。資格を持つ 2 人のユーザー、すなわち主体が同じ DES 鍵を知っている場合、その鍵を使用してテキストを暗号化または復号化することによって、プライベートに通信できます。DES は比較的高速な暗号化メカニズムです。DES チップは暗号化をより高速にします。ただし、チップがなくても、ソフトウェアで代用できます。

DES 鍵を使用する上での問題点は、同じ鍵で暗号化された多数のテキストメッセージを侵入者が収集することによって、鍵が発見されてメッセージが解読される危険性があるということです。このため、Secure NFS などのセキュリティーシステムは鍵を頻繁に変更する必要があります。

Kerberos 認証

Kerberos は、マサチューセッツ工科大学 (MIT) で開発された認証システムです。Kerberos には DES に基づく暗号化もあります。Kerberos V4 は、Secure RPC ではサポートされていません。クライアント側とサーバー側には、RPCSEC_GSS を使用する Kerberos V5 がこのリリースに実装されています。詳細は、第 21 章「Kerberos サービスについて」を参照してください。

Diffie-Hellman 認証と Secure RPC

Diffie-Hellman (DH) のユーザー認証方式は簡単には破られません。クライアントとサーバーは、独自の非公開鍵と公開鍵を使って共通鍵を作り出します。非公開鍵は秘密鍵とも呼ばれます。クライアントとサーバーは、共通鍵を使って相互に通信します。共通鍵は、相互に合意した暗号化機能 (DES など) によって暗号化されます。

認証では、送信側のシステムの共通鍵を使用して現在の時刻を暗号化する機能を利用します。受信側のシステムは、その現在の時刻を復号し、自分の時刻と照合します。クライアントとサーバーで時刻が同期している必要があります。詳細については、『Solaris のシステム管理 (ネットワークサービス)』の「NTP の管理 (作業)」を参照してください。

公開鍵と非公開鍵は、NIS または NIS+ のデータベースに格納されます。NIS では、これらの鍵を publickey マップに格納します。NIS+ では、cred テーブルに格納します。これらのファイルには、すべてのユーザーの公開鍵と非公開鍵が入っています。

システム管理者は、NIS マップまたは NIS+ のテーブルを設定して、ユーザーごとに公開鍵と非公開鍵を生成する必要があります。非公開鍵は、ユーザーのパスワードで暗号化されて格納されます。これにより、その非公開鍵はそのユーザーだけが知っていることになります。

Diffie-Hellman 認証の実装

この節では、Diffie-Hellman 認証 (AUTH_DH) を使用するクライアント
サーバーセッションにおける一連のトランザクションを説明します。

Secure RPC 用の公開鍵と秘密鍵を生成

システム管理者は、認証を開始する前に、newkey または nisaddcred コマンドを実行して公開鍵と秘密鍵を生成します。ユーザーごとに一意の公開鍵と秘密鍵が与えられます。公開鍵は、公開データベースに格納されます。秘密鍵は、暗号化された形式で、同じデータベースに格納されます。chkey コマンドは、公開鍵と秘密鍵のペアを変更します。

Secure RPC 用に keylogin コマンドを実行する

通常、ログインパスワードは Secure RPC パスワードと同じです。この場合、keylogin コマンドは必要ありません。ただし、これらのパスワードが異なる場合は、ユーザーはログインしたあとに keylogin コマンドを実行する必要があります。

keylogin コマンドを入力すると、Secure RPC パスワードの入力を求めるプロンプトが表示されます。コマンドは、このパスワードを使って秘密鍵を復号化します。次に keylogin コマンドは、復号化された秘密鍵をキーサーバープログラムに渡します。キーサーバーは、各コンピュータ上でローカルインスタンスを伴う RPC サービスです。キーサーバーは、復号化された秘密鍵を格納し、ユーザーとサーバーが Secure RPC トランザクションを開始するのを待機します。

ログインパスワードと RPC パスワードが一致している場合は、ログインプロセスは秘密鍵をキーサーバーに渡します。これらのパスワードが異なる場合は、ユーザーが常に keylogin コマンドを実行する必要があります。keylogin コマン

ドをユーザーの環境構成ファイル (~/.login、 ~/.cshrc、 ~/.profile ファイルなど) に設定すると、ユーザーがログインしたときに、keylogin コマンドが自動的に実行されます。

Secure RPC 用の対話鍵の生成

ユーザーがサーバーとトランザクションを開始すると、次の処理が行われます。

1. キーサーバーはランダムに対話鍵を生成します。
2. カーネルは、この対話鍵などを使用してクライアントのタイムスタンプを暗号化します。
3. キーサーバーは、公開鍵データベースからサーバーの公開鍵を検索します。詳細は、publickey(4) のマニュアルページを参照してください。
4. キーサーバーはクライアントの秘密鍵とサーバーの公開鍵を使用して、共通鍵を作成します。
5. キーサーバーは共通鍵を使用して対話鍵を暗号化します。

Secure RPC におけるサーバーとの最初の通信

次に、暗号化したタイムスタンプと暗号化した対話鍵を含む伝送データがサーバーに送信されます。伝送データには資格とベリファイアが含まれます。資格は、次の 3 つの構成要素を持ちます。

- クライアントのネットワーク名
- 共通鍵で暗号化された対話鍵
- 対話鍵で暗号化された「ウィンドウ」

この場合の「ウィンドウ」とは、サーバーの時刻とクライアントのタイムスタンプとの間で許容される時間差のことで、クライアントが指定します。サーバーの時刻とクライアントのタイムスタンプとの間の差がウィンドウより大きい場合、サーバーはクライアントの要求を拒否します。通常の状態では、クライアントは RPC セッションを開始する前にサーバーと同期を取るため、クライアントの要求は拒否されません。

クライアントベリファイアは、次の要素で構成されます。

- 暗号化されたタイムスタンプ
- 指定したウィンドウから 1 を引いた値の暗号化されたベリファイア

ウィンドウベリファイアは、他人がユーザーになりすますのを防ぐために使用されます。なりすましを試みる人は、資格やベリファイアの暗号化された各フィールドに正しい情報の代わりにランダムなビットを挿入するプログラムを作成します。サーバーはこの対話鍵を任意のランダム鍵に復号化し、それを使用してウィンドウとタイムスタンプを復号化しようと試みます。結果は、乱数が生成されるだけです。しかし、数千回の試行を重ねるうちには、このランダムなウィンドウとタイ

ムスタンプのペアが認証システムを通過することが十分ありえます。ウィンドウベリファイアは、偽の資格が認証される可能性を小さくします。

Secure RPC における対話鍵の復号化

サーバーがクライアントから伝送データを受信すると、次の処理が行われます。

1. キーサーバーは、公開鍵データベース内でクライアントの公開鍵を検索します。
2. キーサーバーはクライアントの公開鍵とサーバーの秘密鍵を使用して、共通鍵を計 算します。この共通鍵はクライアントが計算したものと同じです。共通鍵の計算は、秘密鍵の 1 つを知っている必要があるため、そのサーバーとクライアント以外は計算できません。
3. カーネルは共通鍵を使用して、対話鍵を復号化します。
4. カーネルはキーサーバーを呼び出して、復号化された対話鍵によりクライアントのタイムスタンプを復号化します。

Secure RPC におけるサーバーへの情報の格納

サーバーは、クライアントのタイムスタンプを復号化したあと、次の 4 つの情報を資格テーブルに格納します。

- クライアントのコンピュータ名
- 対話鍵
- ウィンドウ
- クライアントのタイムスタンプ

サーバーは、最初の 3 つの情報を将来の使用のために格納します。サーバーはクライアントのタイムスタンプを格納して、同じタイムスタンプが再度使用できないようにします。サーバーは、最後に参照したタイムスタンプよりも時間的にあとのタイムスタンプだけを受け付けます。結果として、すでに届いたトランザクションと同じタイムスタンプを持つトランザクションはすべて拒否されることが保証されます。

注 – このトランザクションにおいて暗黙的に仮定されているのは呼び出し側の名前であり、何らかの方法でこの名前を認証する必要があります。キーサーバーは、呼び出し側を認証するときに、DES 認証を使用できません。キーサーバーが DES 認証を使用すると、デッドロックが発生するためです。キーサーバーは、ユーザー ID (UID) ごとに秘密鍵を格納し、ローカルの `root` のプロセスへの要求だけを認可することによってデッドロックを回避します。

Secure RPC でベリファイアをクライアントに返す

サーバーは、ベリファイアをクライアントに返します。ベリファイアには、次の構成要素が含まれます。

- サーバーが自分の資格キャッシュに記録するインデックス ID
- クライアントのタイムスタンプから 1 を引いた値。対話鍵によって暗号化されます

クライアントのタイムスタンプから 1 を引くのは、タイムスタンプを期限切れにするためです。期限切れのタイムスタンプはクライアントのベリファイアとして再利用できません。

Secure RPC におけるサーバーの認証

クライアントがベリファイアを受信し、そのサーバーを認証します。クライアントは、このベリファイアを送信できるのはサーバーだけであることを知っています。その理由は、クライアントが送信したタイムスタンプの内容を知っているのはサーバーだけだからです。

Secure RPC におけるトランザクションの処理

2 番目以降のすべてのトランザクションごとに、クライアントは次のトランザクションでインデックス ID をサーバーに返します。クライアントは、もう 1 つの暗号化されたタイムスタンプを送信します。サーバーは、クライアントのタイムスタンプから 1 を引いた値を対話鍵で暗号化して、返信します。

Secure RPC の管理 (作業マップ)

次の作業マップでは、NIS、 NIS+、 および NFS に対して Secure RPC を構成する手順を示します。

作業	説明	参照先
1. キーサーバーを起動します。	ユーザーが認証されるために鍵を作成できるようにします。	337 ページの「Secure RPC キーサーバーを再起動する方法」
2. NIS+ ホストで資格を設定します。	NIS+ 環境でホスト上の `root` ユーザーが認証されるようにします。	337 ページの「NIS+ ホストに Diffie-Hellman 鍵を設定する方法」
3. NIS+ ユーザーに鍵を指定します。	NIS+ 環境でユーザーが認証されるようにします。	339 ページの「NIS+ ユーザーに Diffie-Hellman 鍵を設定する方法」
4. NIS ホストで資格を設定します。	NIS 環境でホスト上の `root` ユーザーが認証されるようにします。	339 ページの「NIS ホストに Diffie-Hellman 鍵を設定する方法」

作業	説明	参照先
5. NIS ユーザーに鍵を指定します。	NIS 環境でユーザーが認証されるようにします。	340 ページの「NIS ユーザーに Diffie-Hellman 鍵を設定する方法」
6. 認証によって NFS ファイルを共有します。	NFS サーバーが認証によって共有ファイルシステムを安全に保護できるようにします。	341 ページの「Diffie-Hellman 認証で NFS ファイルを共有する方法」

Secure RPC による認証の管理 (手順)

マウントされた NFS ファイルシステムの使用に対する認証を要求することにより、ネットワークのセキュリティーが増します。

▼ Secure RPC キーサーバーを再起動する方法

1 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 **keyserv** デーモンが動作していることを検証します。

```
# svcs \*keyserv\*
STATE    STIME    FMRI
disabled Dec_14   svc:/network/rpc/keyserv
```

3 キーサーバーサービスがオンラインになっていない場合は、サービスを有効にします。

```
# svcadm enable network/rpc/keyserv
```

▼ NIS+ ホストに Diffie-Hellman 鍵を設定する方法

この手順は、NIS+ ドメインのすべてのホストで実行します。`root` が `keylogin` コマンドを実行すると、サーバーは`mech_dh`に対して GSS-API のアクセプタの資格をもち、クライアントは GSS-API のイニシエータの資格を持ちます。

NIS+ セキュリティーの詳細については、『Solaris のシステム管理 (ネーミングとディレクトリサービス : NIS+ 編)』を参照してください。

1 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 ネームサービスの **publickey** テーブルを有効にします。

次の行を /etc/nsswitch.conf ファイルに追加します。

```
publickey: nisplus
```

3 NIS+ クライアントを起動します。

```
# nisinit -cH hostname
```

hostname は、そのテーブルにクライアントシステム用のエントリを持つ、信頼されている NIS+ サーバー名です。

4 クライアントを **cred** テーブルに追加します。

次のコマンドを入力します。

```
# nisaddcred local
# nisaddcred des
```

5 **keylogin** コマンドを使用して、設定を確認します。

パスワードを求めるプロンプトが出力されたら、この手順は成功です。

```
# keylogin
Password:
```

例 16–1 NIS+ クライアント上で root の新しい鍵を設定する

次の例は、ホスト pluto を使用して、earth を NIS+ クライアントとして設定しています。警告は無視できます。keylogin コマンドが受け付けられて、earth がセキュリティー保護された NIS+ クライアントとして正しく設定されていることを確認しています。

```
# nisinit -cH pluto
NIS Server/Client setup utility.
This system is in the example.com. directory.
Setting up NIS+ client ...
All done.
# nisaddcred local
# nisaddcred des
DES principal name : unix.earth@example.com
Adding new key for unix.earth@example.com (earth.example.com.)
Network password:    <Type password>
Warning, password differs from login password.
Retype password:     <Retype password>
# keylogin
Password:            <Type password>
#
```

▼ NIS+ ユーザーに Diffie-Hellman 鍵を設定する方法

この手順は、NIS+ ドメインのすべてのユーザーで実行します。

1 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 ユーザーをルートマスターサーバー上の **cred** テーブルに追加します。

次のコマンドを入力します。

```
# nisaddcred -p unix.UID@domain-name -P username.domain-name. des
```

この場合、*username.domain-name* の終わりにピリオド (.) を付けてください。

3 クライアントとしてログインし、**keylogin** コマンドを入力して、設定を確認します。

例 16–2 NIS+ ユーザー用の新しい鍵を設定する

次の例では、Diffie-Hellman 認証用の鍵をユーザー jdoe に与えます。

```
# nisaddcred -p unix.1234@example.com -P jdoe.example.com. des
DES principal name : unix.1234@example.com
Adding new key for unix.1234@example.com (jdoe.example.com.)
Password:           <Type password>
Retype password:    <Retype password>
# rlogin rootmaster -l jdoe
% keylogin
Password:           <Type password>
%
```

▼ NIS ホストに Diffie-Hellman 鍵を設定する方法

この手順は、NIS ドメインのすべてのホストで実行します。

1 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 ネームサービスの **publickey** マップを有効にします。

次の行を /etc/nsswitch.conf ファイルに追加します。

```
publickey: nis
```

3 **newkey** コマンドを使用して、新しい鍵のペアを作成します。

```
# newkey -h hostname
```

hostname は、クライアント名です。

例 16–3 NIS クライアント上で root の新しい鍵を設定する

次の例では、earth をセキュリティー保護された NIS クライアントとして設定します。

```
# newkey -h earth
Adding new key for unix.earth@example.com
New Password:       <Type password>
Retype password:    <Retype password>
Please wait for the database to get updated...
Your new key has been successfully stored away.
#
```

▼ NIS ユーザーに Diffie-Hellman 鍵を設定する方法

この手順は、NIS ドメインの各ユーザーに対して実行します。

始める前に システム管理者だけが NIS マスターサーバーにログインしたときに、ユーザーの新しい鍵を作成できます。

1 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 ユーザーの新しい鍵を作成します。

```
# newkey -u username
```

username はユーザー名です。システムはパスワードを求めるプロンプトを出します。汎用パスワードを入力できます。非公開鍵は、汎用パスワードを使用して暗号化されて格納されます。

3 ユーザーに、ログインして **chkey -p** コマンドを入力するように伝えます。

このコマンドでは、そのユーザーは自分だけが知っているパスワードを使用して、自分の非公開鍵を暗号化し直すことができます。

注 – chkey コマンドを使用すると、新しい鍵のペアをユーザーに作成できます。

例 16–4 NIS で新しいユーザー鍵を設定して暗号化する

この例では、スーパーユーザーが鍵を設定します。

```
# newkey -u jdoe
Adding new key for unix.12345@example.com
New Password:        <Type password>
Retype password:     <Retype password>
Please wait for the database to get updated...
Your new key has been successfully stored away.
#
```

次に、ユーザー jdoe が非公開パスワードで鍵を再暗号化します。

```
% chkey -p
Updating nis publickey database.
Reencrypting key for unix.12345@example.com
Please enter the Secure-RPC password for jdoe:     <Type password>
Please enter the login password for jdoe:          <Type password>
Sending key change request to centralexample...
```

▼ Diffie-Hellman 認証で NFS ファイルを共有する方法

この手順では、アクセスに対する認証を要求することにより、NFS サーバー上で共有されているファイルシステムを保護します。

始める前に Diffie-Hellman の公開鍵認証がネットワークで有効である必要があります。ネットワークで認証を有効にするには、次のいずれかを行います。

- 337 ページの「NIS+ ホストに Diffie-Hellman 鍵を設定する方法」
- 339 ページの「NIS ホストに Diffie-Hellman 鍵を設定する方法」

1. スーパーユーザーになるか、**File System Management** プロファイルを含む役割を引き受けます。

   System Administrator 役割には、File System Management プロファイルが含まれます。役割を作成してユーザーに役割を割り当てる場合は、210 ページの「RBAC の構成 (作業マップ)」 を参照してください。

2. **NFS** サーバーで、 **Diffie-Hellman** 認証でファイルシステムを共有します。

   ```
   # share -F nfs -o sec=dh /filesystem
   ```

   filesystem は、共有されているファイルシステムです。

   -o sec=dh オプションは、ファイルシステムにアクセスするために AUTH_DH 認証が必要になるということです。

3 **NFS クライアントで、Diffie-Hellman 認証でファイルシステムをマウントします。**

```
# mount -F nfs -o sec=dh server:filesystem mount-point
```

server *filesystem* を共有しているシステムの名前です

filesystem `/opt` など、共有されているファイルシステムの名前です

mount-point `/opt` など、マウントポイントの名前です

`-o sec=dh` オプションにより、`AUTH_DH` 認証を使ってファイルシステムがマウントされます。

◆ ◆ ◆ 第 17 章

17 PAMの使用

この章では、プラグイン可能認証モジュール (PAM) フレームワークについて説明します。PAM は、Oracle Solaris OS への認証サービスに「プラグイン」するための方法を提供します。PAM によって、システムへのアクセス時に複数の認証サービスを使用できるようになります。

PAM (概要)

プラグイン可能認証モジュール (PAM) フレームワークを使用すると、`login`、 `ftp`、`telnet` などのシステムに入るためのサービスを変更しなくても、新しい認証サービスを「プラグイン」できるようになります。また、PAM を使用すると、UNIX ログインを Kerberos などほかのセキュリティーメカニズムと統合できます。また、アカウント、資格、セッション、およびパスワードの管理メカニズムも「プラグイン」できます。

PAM を使用する利点

PAM フレームワークを使用すると、システムに入るためのサービス (`ftp`、 `login`、`telnet`、 `rsh` など) のユーザー認証を構成できます。PAM には次の利点があります。

- 柔軟な構成ポリシー
  - アプリケーションごとの認証ポリシー
  - デフォルトの認証メカニズムを選択する機能
  - 高度なセキュリティーシステムにおいて複数の承認を要求する機能
- 一般ユーザーにも使いやすい

  - 認証サービスが異なってもパスワードが同じであれば、パスワードを再入力する必要がない
  - ユーザーが複数のコマンドを入力しなくても、複数の認証方式のパスワードを求めるプロンプトを表示できる
- 任意のオプションをユーザー認証サービスに渡す機能
- システムに入るためのサービスを変更しなくても、サイト固有のセキュリティーポリシーを実装する機能

PAM フレームワークの概要

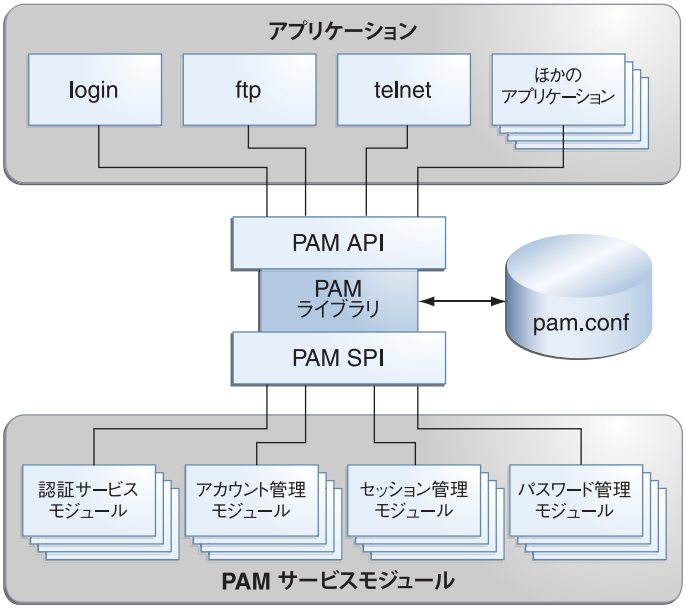
PAM フレームワークは次の 4 つの部分から成ります。

- PAM コンシューマ
- PAM ライブラリ
- pam.conf(4) 構成ファイル
- PAM サービスモジュール。プロバイダとも呼ばれる

このフレームワークは、認証関連アクティビティーの統一的な実施手段を提供します。このアプローチを使えば、アプリケーション開発者は、PAM サービスのポリシーの意味を知らなくてもサービスを使用できるようになります。アルゴリズムは一元的に提供されます。アルゴリズムの変更は、個々のアプリケーションとは無関係に行えます。PAM を使えば、管理者は、アプリケーションを変更しないで、特定システムのニーズに合わせて認証プロセスを調整できるようになります。この調整は、PAM 構成ファイル pam.conf を通じて行われます。

次の図は、PAM のアーキテクチャーを示したものです。アプリケーションは、PAM アプリケーションプログラミングインタフェース (API) 経由で PAM ライブラリと通信します。PAM モジュールは、PAM サービスプロバイダインタフェース (SPI) 経由で PAM ライブラリと通信します。したがって、PAM ライブラリを使えば、アプリケーションとモジュールとの相互通信を実現できます。

図 17–1　PAM のアーキテクチャー



Solaris 10 リリースにおける PAM への変更

Solaris 10 リリースには、プラグイン可能認証モジュール (PAM) フレームワークに対する、次のような変更が含まれています。

- `pam_authtok_check` モジュールが、`/etc/default/passwd` ファイルの新しい調整可能なパラメータを使用して、厳しいパスワードチェックができるようになりました。新しいパラメータは次を定義します。
  - パスワードの共通辞書ワードをチェックするために使用する、コンマで区切られた辞書ファイルのリスト
  - 新しいパスワードと古いパスワードの間に必要な最小限の差異
  - 新しいパスワードで使用する必要がある英字または英字以外の最小文字数
  - 新しいパスワードで使用する必要がある大文字または小文字の最小文字数
  - 許容できる連続的に繰り返される文字の数
- `pam_unix_auth` モジュールが、ローカルユーザーに対するアカウントロックを実装しています。アカウントロックは、`/etc/security/policy.conf` の `LOCK_AFTER_RETRIES` パラメータおよび `/etc/user_attr` の `lock_after_retries` 鍵によって有効になります。詳細は、`policy.conf(4)` および `user_attr(4)` のマニュアルページを参照してください。

- 新しいbinding 制御フラグが定義されました。この制御フラグについては、pam.conf(4) のマニュアルページおよび350 ページの「PAM スタックのしくみ」に記載されています。
- pam_unix モジュールが削除され、同等またはそれ以上の機能を備えた一連のサービスモジュールで置き換えられました。これらのモジュールの多くは、Solaris 9 リリースで導入されました。置き換え後のモジュールのリストは、次のとおりです。
  - pam_authtok_check
  - pam_authtok_get
  - pam_authtok_store
  - pam_dhkeys
  - pam_passwd_auth
  - pam_unix_account
  - pam_unix_auth
  - pam_unix_cred
  - pam_unix_session
- 以前の pam_unix_auth モジュールの機能は、2 つのモジュールに分割されました。pam_unix_auth モジュールは、ユーザーのパスワードが正しいかどうかを検証するように変更されました。新しく追加された pam_unix_cred モジュールは、ユーザーの資格情報を確立する機能を提供します。
- pam_krb5 モジュールに、PAM フレームワークを使って Kerberos 資格キャッシュを管理する機能が追加されました。
- 新しい pam_deny モジュールが追加されました。このモジュールは、サービスへのアクセスを拒否するために使用できます。デフォルトでは、pam_deny モジュールは無効になっています。詳細は、pam_deny(5) のマニュアルページを参照してください。

PAM (手順)

この節では、PAM のフレームワークが特定のセキュリティーポリシーを使用するために必要な作業について説明します。PAM 構成ファイルに関連するセキュリティーのいくつかの問題について注意する必要があります。セキュリティーの問題については、347 ページの「PAM の実装計画」を参照してください。

PAM (作業マップ)

作業	説明	参照先
PAM のインストールを計画します。	ソフトウェア構成処理を開始する前に、構成を検討および決定します。	347 ページの「PAM の実装計画」

作業	説明	参照先
新しい PAM モジュールを追加します。	必要に応じて、サイト固有のモジュールを作成およびインストールし、汎用ソフトウェアにない要件に対応します。この手順ではこれらの新しい PAM モジュールのインストール方法について説明します。	348 ページの「PAM モジュールを追加する方法」
`~/.rhosts` によるアクセスを拒否します。	`~/.rhosts` によるアクセスを拒否して、セキュリティーを強化します。	349 ページの「PAM を使用して、遠隔システムからの rhost 式アクセスを防ぐ方法」
エラー記録を開始します。	`syslog` を使用して PAM エラーメッセージの記録を開始します。	349 ページの「PAM のエラーレポートを記録する方法」

PAM の実装計画

配布時に、`pam.conf` 構成ファイルは標準的なセキュリティーポリシーを実装します。このポリシーは、多くの状況で機能します。別のセキュリティーポリシーを実装する必要がある場合に注意すべき問題は、次のとおりです。

- 何が必要か、特にどの PAM サービスモジュールを選択するかを決定します。
- 特別な構成オプションが必要なサービスを確認します。適宜、`other` を使用します。
- モジュールを実行する順番を決定します。
- 各モジュールに対する制御フラグを選択します。すべての制御フラグの詳細については、350 ページの「PAM スタックのしくみ」を参照してください。
- 各モジュールに必要なオプションを選択します。各モジュールのマニュアルページには、特別なオプションが一覧表示されます。

ここで、PAM 構成ファイルを変更する前に次のことを考慮することをお勧めします。

- `/etc/pam.conf` ですべてのアプリケーションを指定しなくてもいいように、モジュールタイプごとに `other` エントリを使用します。
- `binding`、 `sufficient`、 および `optional` 制御フラグのセキュリティーへの影響を確認します。
- モジュールに関連するマニュアルページを参照します。これらのマニュアルページには、各モジュールの機能、使用可能なオプション、スタック内のモジュール間の相互作用などの説明があります。

注意 – PAM 構成ファイルの構成を間違えたり壊したりすると、どのユーザーもログインできなくなる可能性があります。この場合、sulogin コマンドは PAM を使用しないので、root パスワードを使用してマシンをシングルユーザーモードでブートして問題を解決する必要があります。

/etc/pam.conf ファイルを変更したあと、システムにアクセスできる間にできる限り調査して問題を解決します。変更によって影響を受ける可能性があるコマンドは、すべてテストしてください。たとえば、新しいモジュールを telnet サービスに追加した場合、telnet コマンドを使用して、行った変更が期待どおりに動作しているかどうかを検証します。

▼ PAM モジュールを追加する方法

この手順では、新しい PAM モジュールを追加する方法を示します。新しいモジュールは、サイト固有のセキュリティーポリシーを網羅するためや、Sun 以外のアプリケーションをサポートするために作成します。

1. スーパーユーザーになるか、同等の役割を引き受けます。

   役割には、認証と特権コマンドが含まれます。役割の詳細は、210 ページの「RBAC の構成 (作業マップ)」を参照してください。

2. 使用する制御フラグとオプションを決定します。

   制御フラグについては、350 ページの「PAM スタックのしくみ」を参照してください。

3. モジュールファイルの所有者が **root** で、そのアクセス権が **555** になるように設定します。

4. **PAM** 構成ファイル **/etc/pam.conf** を編集して、このモジュールを適切なサービスに追加します。

5. モジュールが適切に追加されたことを検証します。

   構成ファイルが間違って構成されているおそれもあるので、システムをリブートする前にテストを行う必要があります。システムをリブートする前に、ssh などの直接サービスを使用してログインし、su コマンドを実行します。サービスは、システムをブートしたときに 1 度だけ生成されるデーモンである場合があります。その場合には、システムをリブートしてから、モジュールが追加されていることを確認する必要があります。

▼ PAM を使用して、遠隔システムからの rhost 式アクセスを防ぐ方法

1. スーパーユーザーになるか、同等の役割を引き受けます。

   役割には、認証と特権コマンドが含まれます。役割の詳細は、210 ページの「RBAC の構成 (作業マップ)」を参照してください。

2. **rhosts_auth.so.1** を含む行をすべて **PAM** 構成ファイルから削除します。

   この手順によって、`rlogin` セッション中、`~/.rhosts` ファイルは読み取られなくなります。これにより、ローカルシステムに認証されていない遠隔システムからのアクセスを防止できます。`~/.rhosts` ファイルまたは `/etc/hosts.equiv` ファイルの存在またはその内容にかかわらず、すべての `rlogin` アクセスにはパスワードが必要になります。

3. **rsh** サービスを無効にします。

   `~/.rhosts` ファイルへのその他の非承認アクセスを防ぐには、`rsh` サービスも無効にする必要があります。

   ```
   # svcadm disable network/shell
   ```

▼ PAM のエラーレポートを記録する方法

1. スーパーユーザーになるか、同等の役割を引き受けます。

   役割には、認証と特権コマンドが含まれます。役割の詳細は、210 ページの「RBAC の構成 (作業マップ)」を参照してください。

2. 必要な記録のレベルに **/etc/syslog.conf** ファイルを構成します。

   記録レベルの詳細については、`syslog.conf(4)` を参照してください。

3. **syslog** デーモンの構成情報を再表示します。

   ```
   # svcadm refresh system/system-log
   ```

PAM の構成 (参照)

`login`、 `rlogin`、 `su`、 `cron` などのシステムサービスに対する PAM サービスモジュールを構成するには、PAM 構成ファイル `pam.conf(4)` を使用します。システム管理者がこのファイルを管理します。`pam.conf` 内のエントリの順番が間違っていると、予期しない副作用が生じる可能性があります。たとえば、`pam.conf` の設定が不適切であると、ユーザーがロックアウトされ、修復のためにシングルユーザーモードが必要になる可能性があります。順番の設定方法については、350 ページの「PAM スタックのしくみ」を参照してください。

PAM 構成ファイルの構文

構成ファイル内のエントリの形式は、次のとおりです。

service-name module-type control-flag module-path module-options

service-name サービスの名前。たとえば、`ftp`、`login`、または `passwd` などです。アプリケーションは、自身が提供するサービスごとに異なるサービス名を使用できます。たとえば、Oracle Solaris Secure Shell デーモンが使用するサービス名は次のとおりです。`sshd-none`、`sshd-password`、`sshd-kbdint`、`sshd-pubkey`、`sshd-hostbased`。サービス名 *other* は事前に定義された名前であり、ワイルドカードのサービス名として使用されます。構成ファイル内で特定のサービス名が見つからない場合には、*other* の構成が使用されます。

module-type サービスのタイプ。具体的には `auth`、`account`、`session`、`password` のいずれかです。

control-flag サービスの統合された成功または失敗の値を決定するうえで、そのモジュールの果たす役割を示します。有効な制御フラグは、`binding`、`include`、`optional`、`required`、`requisite`、および `sufficient` です。制御フラグの使用方法については、350 ページの「PAM スタックのしくみ」を参照してください。

module-path サービスを実装しているライブラリオブジェクトへのパス。パス名が絶対パスでない場合、そのパス名は `/usr/lib/security/$ISA/` に対する相対パスとみなされます。`libpam` による検索が、アプリケーションの特定のアーキテクチャーに対するディレクトリ内で行われるように、アーキテクチャーに依存するマクロ $ISA を使用します。

module-options サービスモジュールに渡されるオプション。各モジュールのマニュアルページには、そのモジュールで使用できるオプションの説明があります。典型的なモジュールオプションとしては、`nowarn` や `debug` などがあります。

PAM スタックのしくみ

あるアプリケーションが次の関数を呼び出すと、`libpam` は構成ファイル `/etc/pam.conf` を読み取り、そのサービスの操作に関与するモジュールを決定します。

- `pam_authenticate(3PAM)`
- `pam_acct_mgmt(3PAM)`
- `pam_setcred(3PAM)`

- pam_open_session(3PAM)
- pam_close_session(3PAM)
- pam_chauthtok(3PAM)

認証管理やアカウント管理など、このサービスのある操作に対するモジュールが /etc/pam.conf に 1 つしか含まれていない場合、そのモジュールの結果によって、その操作の結果が決まります。たとえば、passwd アプリケーションのデフォルトの認証操作には、1 つのモジュール pam_passwd_auth.so.1 が含まれています。

```
passwd  auth required           pam_passwd_auth.so.1
```

これに対し、サービスのある操作に対して複数のモジュールが定義されている場合、それらのモジュールは「スタック」を形成しており、そのサービスに対する「PAM スタック」が存在している、と言います。たとえば、pam.conf に次のエントリが含まれる場合を考えます。

```
login   auth requisite          pam_authtok_get.so.1
login   auth required           pam_dhkeys.so.1
login   auth required           pam_unix_cred.so.1
login   auth required           pam_unix_auth.so.1
login   auth required           pam_dial_auth.so.1
```

これらのエントリは、login サービスに対するサンプルの auth スタックを表現したものです。このスタックの結果を決定するには、個々のモジュールの結果コードに対して「統合プロセス」を実行する必要があります。統合プロセスでは、各モジュールが、/etc/pam.conf 内で指定された順番で実行されます。個々の成功コードまたは失敗コードが、モジュールの制御フラグに基づいて総合結果へと統合されます。制御フラグによっては、スタックの早期終了が発生する可能性があります。たとえば、requisite モジュールが失敗したり、sufficient モジュールや binding モジュールが成功したりする可能性があります。スタックの処理が完了すると、個々の結果が組み合わされて単一の総合結果が算出され、それがアプリケーションに渡されます。

制御フラグは、サービスへのアクセスを決定するうえで、ある PAM モジュールが果たす役割を示します。制御フラグとその効果は、次のとおりです。

- **binding** – binding モジュールの要件を満たすことに成功すると、その時点までの required モジュールがどれも失敗しなかった場合には、すぐにアプリケーションに成功が返されます。これらの条件が満たされると、後続のモジュールは実行されません。失敗すると、required 失敗が記録され、モジュールの処理が継続されます。
- **include** – 別の PAM 構成ファイルに含まれる行を追加し、それらの行がその時点で PAM スタック内で使用されるようにします。このフラグは成功または失敗の動作を制御しません。新しいファイルが読み取られると、PAM のインクルードスタックが 1 増やされます。新しいファイル内でのスタックチェックが完了すると、インクルードスタック値が 1 減らされます。ファイルの末尾に到達し、かつ PAM インクルードスタックが 0 であれば、スタック処理は終了します。PAM インクルードスタックの最大数は、32 です。
- **optional** – optional モジュールの要件を満たすことに成功することは、サービスを使用するために必要な条件ではありません。失敗すると、optional 失敗が記録されます。
- **required** – required モジュールの要件を満たすことに成功することは、サービスを使用するために必要な条件です。失敗すると、このサービスの残りのモジュールの実行完了後に、エラーが返されます。サービスの最終的な成功が返されるのは、binding または required モジュールがどれも失敗を報告しなかった場合だけです。
- **requisite** – requisite モジュールの要件を満たすことに成功することは、サービスを使用するために必要な条件です。失敗するとすぐにエラーが返され、後続のモジュールは実行されません。あるサービスのすべての requisite モジュールが成功を返さないと、この機能がアプリケーションに成功を返すことができません。
- **sufficient** – その時点までに required 失敗が一度も発生しなかった場合、sufficient モジュールが成功するとすぐにアプリケーションに成功が返され、後続のモジュールは実行されません。失敗すると、optional 失敗が記録されます。

次の 2 つの図は、統合プロセスでアクセスがどのように決定されるかを示したものです。最初の図は、制御フラグのタイプごとに成功または失敗がどのように記録されるのかを示しています。2 番目の図は、統合された値の決定方法を示しています。

図 17–2　PAM スタック: 制御フラグの効果

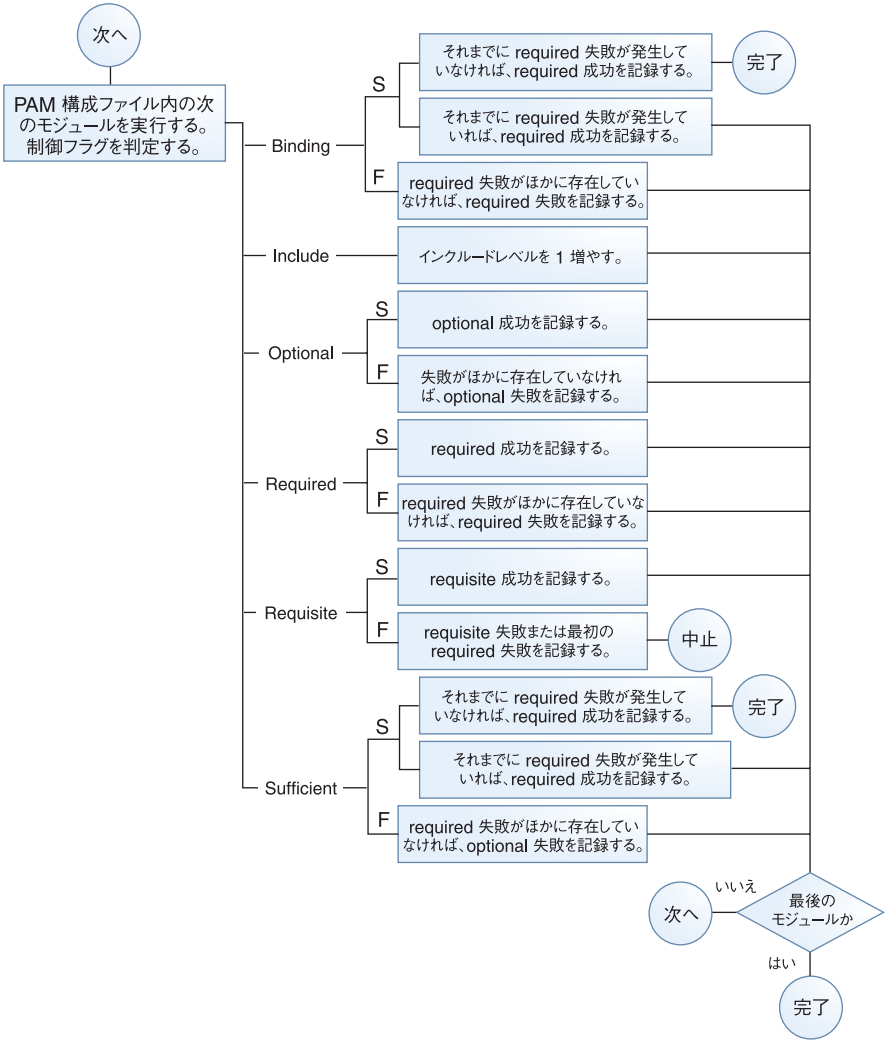
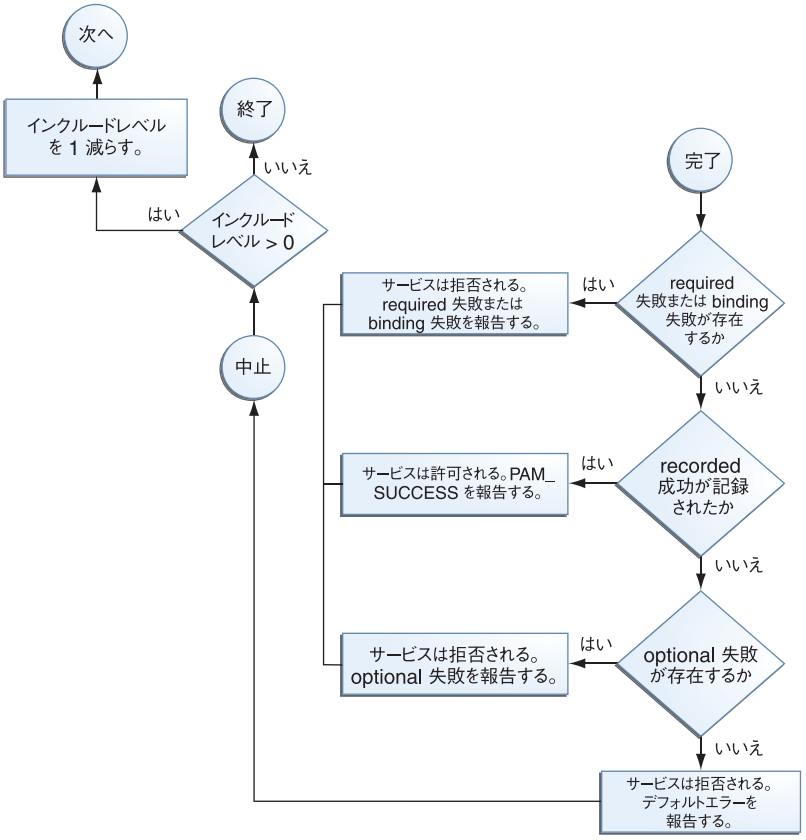


図 17–3　PAM スタック:統合された値の決定方法



PAM スタックの例

次のような、認証を要求する `rlogin` サービスの例を考えます。

例 17–1　典型的な PAM 構成ファイルの内容の一部

この例の `pam.conf` ファイルには、`rlogin` サービスに対して次のような内容が含まれています。

```
# Authentication management
...
# rlogin service
rlogin  auth sufficient         pam_rhosts_auth.so.1
rlogin  auth requisite          pam_authtok_get.so.1
rlogin  auth required           pam_dhkeys.so.1
rlogin  auth required           pam_unix_auth.so.1
...
```

例 17–1　典型的な PAM 構成ファイルの内容の一部　　(続き)

rlogin サービスが認証を要求すると、libpam はまず、pam_rhosts_auth(5) モジュールを実行します。pam_rhosts_auth モジュールの制御フラグは、sufficient に設定されています。pam_rhosts_auth モジュールがユーザーの認証に成功すると、処理が中止され、アプリケーションに成功が返されます。

pam_rhosts_auth モジュールがユーザーの認証に失敗すると、次の PAM モジュールである pam_authtok_get(5) が実行されます。このモジュールの制御フラグは、requisite に設定されています。pam_authtok_get が失敗すると、認証処理が終了し、失敗が rlogin に返されます。

pam_authtok_get が成功すると、次の 2 つのモジュール pam_dhkeys(5) と pam_unix_auth(5) が実行されます。どちらのモジュールにも、required に設定された制御フラグが関連付けられているため、各モジュールが失敗を返すかどうかにかかわらず、処理が継続されます。pam_unix_auth の実行が終了すると、rlogin 認証用のモジュールはもう残っていません。この時点で、pam_dhkeys、 pam_unix_auth のいずれかが失敗を返していた場合、ユーザーは rlogin 経由でのアクセスを拒否されます。

◆ ◆ ◆ 第 18 章

18 SASLの使用

この章では、簡易認証セキュリティー層 (SASL) について説明します。

SASL (概要)

簡易認証セキュリティー層 (SASL) は、ネットワークプロトコルに認証サービスとセキュリティーサービス (オプション) を提供するフレームワークです。アプリケーションは、SASL ライブラリ `/usr/lib/libsasl.so` を呼び出します。SASL ライブラリは、アプリケーションと各種 SASL メカニズムを結び付ける層の役割を果たします。このメカニズムは、認証プロセスや、オプションのセキュリティーサービスの提供時に使用されます。SASL のバージョンは Cyrus SASL から派生したものですが、少し変更されています。

SASL は、次のサービスを提供します。

- プラグインを読み込む
- アプリケーションから必要なセキュリティーオプションを判断してセキュリティーメカニズムの選択を支援する
- アプリケーションで使用可能なプラグインを一覧表示する
- 特定の認証の試みに対して、使用可能なメカニズムの一覧から最適なメカニズムを選択する
- アプリケーションと選択されたメカニズムの間で認証データの経路を指定する
- アプリケーションに返す SASL ネゴシエーションに関する情報を提供する

SASL (参照)

次の節では、SASL の実装についての情報を提供します。

SASL プラグイン

SASL プラグインは、セキュリティーメカニズム、ユーザーの正規化、および補助プロパティーの検索をサポートします。デフォルトでは、動的に読み込まれる 32 ビットのプラグインは `/usr/lib/sasl` にインストールされ、64 ビットのプラグインは `/usr/lib/sasl/`*$ISA* にインストールされます。次のセキュリティーメカニズムプラグインが提供されます。

`crammd5.so.1` CRAM-MD5。認証のみをサポートし、承認はサポートしません

`digestmd5.so.1` DIGEST-MD5。認証、整合性、機密性のほか、承認もサポートします

`gssapi.so.1` GSSAPI。認証、整合性、機密性のほか、承認もサポートします。GSSAPI セキュリティーメカニズムには、機能している Kerberos 基盤が必要です。

`plain.so.1` PLAIN。認証と承認をサポートします。

さらに、 EXTERNAL セキュリティーメカニズムのプラグインと INTERNAL ユーザー正規化プラグインが、`libsasl.so.1`に装備されています。EXTERNAL メカニズムは、認証と承認をサポートします。外部のセキュリティーソースによって提供された場合には、整合性と機密性がサポートされます。INTERNAL プラグインは、必要に応じて、レルム名をユーザー名に追加します。

Oracle Solaris のリリースは、現時点では `auxprop` プラグインを提供していません。CRAM-MD5 および DIGEST-MD5 メカニズムプラグインをサーバー側で完全に機能させるには、ユーザーは `auxprop` プラグインを用意して平文のパスワードを取り出す必要があります。PLAIN プラグインでは、さらに、パスワードを検証できるようにするために追加のサポートが必要です。パスワード検証に利用できるものは、サーバーアプリケーションへのコールバック、`auxprop` プラグイン、`saslauthd`、または `pwcheck` のいずれかです。`salauthd` デーモンおよび `pwcheck` デーモンは、今回の Oracle Solaris のリリースでは提供されません。相互運用性を向上させるために、サーバーアプリケーションは、`mech_list` SASL オプションによって完全に機能するメカニズムに制限してください。

SASL の環境変数

デフォルトでは、クライアントの認証名は `getenv("LOGNAME")` に設定されます。この変数は、クライアントまたはプラグインによってリセットすることができます。

SASL のオプション

libsasl およびプラグインの動作は、/etc/sasl/*app*.conf ファイルで設定可能なオプションを使用してサーバー上で変更することができます。変数 *app* は、サーバー定義のアプリケーション名です。サーバー *app* のマニュアルでは、サーバー定義のアプリケーション名が指定されています。

次のオプションを指定できます。

auto_transition　ユーザーが平文認証に成功すると、自動的にそのユーザーをほかのメカニズムに切り替えます。

auxprop_login　使用する補助プロパティープラグインの名前を列挙します。

canon_user_plugin　使用する canon_user プラグインを選択します。

mech_list　サーバーアプリケーションが使用可能なメカニズムを列挙します。

pwcheck_method　パスワードを検証するために使用するメカニズムを列挙します。現在は、auxprop が唯一使用可能な値です。

reauth_timeout　迅速に再認証するために認証情報がキャッシュされる時間の長さを分単位で設定します。このオプションは、DIGEST-MD5 プラグインで使用されます。このオプションを 0 に設定すると、再認証が無効になります。

次のオプションはサポートされません。

plugin_list　使用可能なメカニズムを列挙します。このオプションは、プラグインの動的な読み込みの動作を変えるため、使用されなくなりました。

saslauthd_path　saslauthd デーモンとの通信に使用される saslauthd ドアの場所を定義します。saslauthd デーモンは、Oracle Solaris リリースに組み込まれていません。このため、このオプションも組み込まれていません。

keytab　GSSAPI プラグインによって使用される keytab ファイルの場所を定義します。代わりに KRB5_KTNAME 環境変数を使用して、デフォルトの keytab の場所を設定します。

次のオプションは、Cyrus SASL では用意されていません。ただし、Oracle Solaris リリースでは追加されています。

use_authid　GSS クライアントセキュリティーコンテキストの作成時に、デフォルトの資格を使用するのではなく、クライアントの資格を取得します。デフォルトでは、デフォルトのクライアント Kerberos 識別情報が使用されます。

`log_level`　　サーバーのログのレベルを設定します。

◆ ◆ ◆ 第 19 章

Oracle Solaris Secure Shell の使用 (手順)

Secure Shell 機能によって、セキュリティー保護されていないネットワークを介して、リモートホストへの安全なアクセスが可能になります。Solaris Secure Shell には、遠隔ログインおよび遠隔ファイル転送を行うコマンドが組み込まれています。この章で説明する内容は次のとおりです。

参照情報については、第 20 章「Oracle Solaris Secure Shell (参照)」を参照してください。Oracle Solaris Secure Shell と OpenSSH プロジェクトの関係については、364 ページの「Oracle Solaris Secure Shell と OpenSSH プロジェクト」を参照してください。

Oracle Solaris Secure Shell (概要)

Secure Shell の認証は、パスワードまたは公開鍵、あるいはその両方を使用して行われます。すべてのネットワークトラフィックは暗号化されます。このため、Secure Shell では、悪意を持つ侵入者が傍受した通信を読むことはできません。また、攻撃者が偽装することもできません。

Secure Shell は、オンデマンドタイプの 仮想プライベートネットワーク (VPN) として使用することもできます。VPN では、暗号化されたネットワークリンクを介して、ローカルマシンと遠隔マシン間で、X ウィンドウシステムのトラフィックを転送したり個々のポート番号を接続したりできます。

Secure Shell では、次の操作を行うことができます。

- セキュリティー保護されていないネットワークを介して、別のホストに安全にログインします。
- 2 つのホスト間でファイルを安全にコピーします。

- 遠隔ホスト上でコマンドを安全に実行します。

Secure Shell は 2 つのバージョンの Secure Shell プロトコルをサポートします。バージョン 1 は、Secure Shell プロトコルのオリジナルバージョンです。バージョン 2 は、安全性が向上し、バージョン 1 の基本的なセキュリティー設計上の欠陥が修正されています。バージョン 1 は、バージョン 2 へ移行するユーザーを支援する目的だけに提供しています。バージョン 1 は、極力使用しないでください。

注 – このドキュメントでは、v1 はバージョン 1、 v2 はバージョン 2 を表しています。

Oracle Solaris Secure Shell 認証

Secure Shell は、遠隔ホストへの接続を認証するために、公開鍵とパスワードの方式を提供します。公開鍵認証は、パスワード認証よりも強力な認証メカニズムです。これは、非公開鍵がネットワーク上を移動しないためです。

認証方式は、次の順序で試されます。構成が認証方式を満たさないときは、次の方式が試されます。

- **GSS-API** –`mech_krb5` (Kerberos V) や `mech_dh` (AUTH_DH) などの GSS-API メカニズムの資格を使用して、クライアントとサーバーを認証します。GSS-API の詳細は、『Oracle Solaris セキュリティーサービス開発ガイド』の「GSS-API の紹介」を参照してください。
- ホストに基づく認証 – ホスト鍵と rhosts ファイルを使用します。クライアントの RSA および DSA 公開/非公開ホスト鍵を使用してクライアントを認証します。rhosts ファイルを使用して、ユーザーに対してクライアントを承認します。
- 公開鍵認証 – RSA および DSA 公開/非公開鍵によってユーザーを認証します。
- パスワード認証 – PAM を使用してユーザーを認証します。v2 でのキーボード認証方式では、PAM による任意のプロンプトが可能です。詳細については、`sshd`(1M) のマニュアルページの「`SECURITY`」のセクションを参照してください。

次の表では、リモートホストにログインしようとするユーザーを認証するための要件を示します。ユーザーは、ローカルホスト (クライアント) 上に存在します。遠隔ホスト (サーバー) は、`sshd` デーモンを実行しています。次の表は、Secure Shell の認証方式、互換性のあるプロトコルのバージョン、およびホストの要件の一覧です。

表 19–1 Secure Shell の認証方式

認証方式 (プロトコルのバージョン)	ローカルホスト (クライアント) の要件	遠隔ホスト (サーバー) の要件
GSS-API (v2)	GSS メカニズムのイニシエータの資格。	GSS メカニズムのアクセプタの資格。詳細は、385 ページの「Secure Shell での GSS 資格の取得」を参照してください。

表 19–1 Secure Shell の認証方式 (続き)

認証方式 (プロトコルのバージョン)	ローカルホスト (クライアント) の要件	遠隔ホスト (サーバー) の要件
ホストに基づく (v2)	ユーザーアカウント /etc/ssh/ssh_host_rsa_key または /etc/ssh/ssh_host_dsa_key にローカルホストの非公開鍵 /etc/ssh/ssh_config 内で HostbasedAuthentication yes	ユーザーアカウント /etc/ssh/known_hosts または ~/.ssh/known_hosts にローカルホストの公開鍵 /etc/ssh/sshd_config 内で HostbasedAuthentication yes /etc/ssh/sshd_config 内で IgnoreRhosts no /etc/ssh/shosts.equiv、 /etc/hosts.equiv、 ~/.rhosts、 または ~/.shosts にローカルホストのエントリ
RSA または DSA 公開鍵 (v2)	ユーザーアカウント ~/.ssh/id_rsa または ~/.ssh/id_dsa に非公開鍵 ~/.ssh/id_rsa.pub または ~/.ssh/id_dsa.pub にユーザーの公開鍵	ユーザーアカウント ~/.ssh/authorized_keys にユーザーの公開鍵
RSA 公開鍵 (v1)	ユーザーアカウント ~/.ssh/identity に非公開鍵 ~/.ssh/identity.pub にユーザーの公開鍵	ユーザーアカウント ~/.ssh/authorized_keys にユーザーの公開鍵
Keyboard-interactive (v2)	ユーザーアカウント	ユーザーアカウント パスワードの有効期限が切れたときの任意のプロンプトやパスワード変更など、PAM をサポートします。
パスワードに基づく (v1 または v2)	ユーザーアカウント	ユーザーアカウント PAM をサポートします。
.rhosts のみ (v1)	ユーザーアカウント	ユーザーアカウント /etc/ssh/sshd_config 内で IgnoreRhosts no /etc/ssh/shosts.equiv、 /etc/hosts.equiv、 ~/.shosts、 または ~/.rhosts にローカルホストのエントリ

表 19–1 Secure Shell の認証方式 (続き)

認証方式 (プロトコルのバージョン)	ローカルホスト (クライアント) の要件	遠隔ホスト (サーバー) の要件
`.rhosts` と RSA (v1) (サーバーのみ)	ユーザーアカウント `/etc/ssh/ssh_host_rsa1_key` にローカルホストの公開鍵	ユーザーアカウント `/etc/ssh/ssh_known_hosts` または `~/.ssh/known_hosts` にローカルホストの公開鍵 `/etc/ssh/sshd_config` 内で `IgnoreRhosts no` `/etc/ssh/shosts.equiv`、 `/etc/hosts.equiv`、 `~/.shosts`、 または `~/.rhosts` にローカルホストのエントリ

企業における Secure Shell

Oracle Solaris システム上の Secure Shell の総合的な説明については、『Secure Shell in the Enterprise』 (Jason Reid 著、ISBN 0-13-142900-0、 2003 年 6 月) を参照してください。このドキュメントは、Sun Microsystems Press によって発行されている Sun BluePrints Series の 1 つです。

Oracle Solaris Secure Shell と OpenSSH プロジェクト

Oracle Solaris Secure Shell は OpenSSH (`http://www.openssh.com`) プロジェクトのフォークです。OpenSSH の最近のバージョンに見つかった脆弱性に対するセキュリティー関連の修正が、個別のバグ修正および機能として Oracle Solaris Secure Shell に組み込まれています。Oracle Solaris Secure Shell フォークに対しては内部開発が継続されます。

Oracle Solaris の技術者はプロジェクトにバグ修正を提供するほか、次の の機能を Secure Shell の Oracle Solaris フォークに組み込みました。

- PAM - Oracle Solaris Secure Shell では PAM が使用されます。OpenSSH の `UsePAM` 設定オプションはサポートされていません。
- 特権の分離 - Oracle Solaris Secure Shell では OpenSSH プロジェクトの特権分離コードは使用されません。Oracle Solaris Secure Shell では、監査、記録管理、および再キーイングの処理がセッションプロトコルの処理から分離されます。

  Oracle Solaris Secure Shell の特権分離コードは常にオンに設定されており、オフに切り替えることはできません。OpenSSH の `UsePrivilegeSeparation` 設定オプションはサポートされていません。
- ロケール - Oracle Solaris Secure Shell では、RFC 4253「Secure Shell Transfer Protocol」で定義されている言語ネゴシエーションが完全にサポートされています。ユーザーがログインしたあと、ユーザーのログインシェルプロファイルを Secure Shell でネゴシエーションを行なったロケール設定に優先することができます。

- 監査 - Oracle Solaris Secure Shell は Oracle Solaris 監査サブシステムに完全に統合されています。監査については、パート VII「Oracle Solaris 監査」を参照してください。
- GSS-API のサポート - GSS-API はユーザー認証と初期鍵交換の両方に使用できます。GSS-API は RFC 4462「Generic Security Service Application Program Interface」で定義されています。
- プロキシコマンド - Oracle Solaris Secure Shell では、SOCKS5 プロトコルと HTTP プロトコルのプロキシコマンドが提供されています。例については、380 ページの「ファイアウォール外部のホストにデフォルト接続を設定する方法」を参照してください。

Solaris 9 リリース以降、Oracle Solaris Secure Shell に次の変更点が取り入れられています。

- Oracle Solaris Secure Shell は OpenSSH 3.5p1 からフォークされます。
- `/etc/ssh/sshd_config` ファイルの `X11Forwarding` のデフォルト値が、`yes` になりました。
- 次のキーワードが採用されました。
  - `GSSAPIAuthentication`
  - `GSSAPIKeyExchange`
  - `GSSAPIDelegateCredentials`
  - `GSSAPIStoreDelegatedCredentials`
  - `KbdInteractiveAuthentication`

  `GSSAPI` キーワードによって、 Oracle Solaris Secure Shell で GSS 資格を認証に使用できます。`KbdInteractiveAuthentication` キーワードが、PAM での任意のプロンプトとパスワードの変更をサポートします。キーワードとそのデフォルト値の一覧については、387 ページの「Secure Shell でのキーワード」を参照してください。
- ARCFOUR 暗号および AES128-CTR 暗号を使用できます。ARCFOUR は RC4 としても知られています。AES 暗号は、カウンタモードの AES です。
- `sshd` デーモンが、`/etc/default/login` および `login` コマンドの変数を使用します。`/etc/default/login` の変数は、`sshd_config` ファイルの値によって無効にすることができます。詳細は、391 ページの「Secure Shell およびログインの環境変数」と `sshd_config(4)` のマニュアルページを参照してください。
- 接続が認証されると、サーバーの `ChrootDirectory` オプションによって、接続されたクライアントをオプションが指定するディレクトリに `chroot` することができます。このオプションはプロセス内 SFTP サーバー、つまり内部 SFTP をサポートし、その構成は `ChrootDirectory` オプションを使用することによって簡素化されています。

Oracle Solaris Secure Shell (作業マップ)

次の作業マップは、Secure Shell の構成と、Oracle Solaris の Secure Shell 機能の使用についての作業マップです。

作業	説明	参照先
Secure Shell を構成します	ユーザー向けに Secure Shell を構成する管理者をガイドします。	366 ページの「Oracle Solaris Secure Shell の構成 (作業マップ)」
Secure Shell を使用します	Secure Shell を使用するユーザーをガイドします。	371 ページの「Oracle Solaris Secure Shell の使用 (作業マップ)」

Oracle Solaris Secure Shell の構成 (作業マップ)

次の作業マップでは、Secure Shell の構成手順を示します。

作業	説明	参照先
ホストに基づく認証を構成します	クライアントとサーバーでのホストに基づく認証を構成します。	366 ページの「ホストに基づく認証を Secure Shell に設定する方法」
v1 および v2 を使用できるようにホストを構成します	v1 および v2 のプロトコルを使用するホストに対して公開鍵のファイルを作成します。	369 ページの「Secure Shell v1 を有効にする方法」
ポート転送を構成します	ユーザーがポート転送を使用できるようにします。	370 ページの「Secure Shell のポート転送を構成する方法」

Oracle Solaris Secure Shell の構成 (手順)

Secure Shell では、デフォルトでは、ホストに基づく認証と両方のプロトコルの使用は無効になっています。これらのデフォルトを変更するには、管理者の介入が必要です。また、ポート転送が機能するようにするためにも、管理者の介入が必要です。

▼ ホストに基づく認証を Secure Shell に設定する方法

次の手順によって公開鍵システムが設定され、クライアントの公開鍵がサーバー上での認証に使用できるようになります。また、ユーザーは、公開鍵と非公開鍵のペアを作成する必要があります。

この手順での「クライアント」および「ローカルホスト」という用語は、ユーザーが ssh コマンドを入力するマシンを指しています。「サーバー」および「リモートホスト」という用語は、クライアントがアクセスを試みるマシンを指しています。

1 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 クライアントで、ホストに基づく認証を有効にします。

クライアントの構成ファイル /etc/ssh/ssh_config で、次のエントリを入力します。

```
HostbasedAuthentication yes
```

このファイルの構文については、ssh_config(4) のマニュアルページを参照してください。

3 サーバーで、ホストに基づく認証を有効にします。

サーバーの構成ファイル /etc/ssh/ssh_config で、同じエントリを入力します。

```
HostbasedAuthentication yes
```

このファイルの構文については、sshd_config(4) のマニュアルページを参照してください。

4 サーバーで、クライアントが信頼されるホストとして認識されるようにするファイルを構成します。

詳細については、sshd(1M) のマニュアルページの「FILES」のセクションを参照してください。

- サーバーの **/etc/ssh/shosts.equiv** ファイルへのエントリとしてクライアントを追加します。

  client-host

- または、クライアント用のエントリをサーバー上の **~/.shosts** ファイルに追加するようにユーザーに指示することもできます。

  client-host

5 サーバーで、**sshd** デーモンが信頼されるホストのリストにアクセスできるようにします。

/etc/ssh/sshd_config ファイルで、IgnoreRhosts を no に設定します。

```
## sshd_config
IgnoreRhosts no
```

6 使用するサイトの **Secure Shell** のユーザーが両方のホストでアカウントを持つようにします。

7 クライアントの公開鍵をサーバー上に置くために、次のどちらかを行います。

- サーバー上の **sshd_config** ファイルを変更後、クライアントの公開ホスト鍵を **~/.ssh/known_hosts** ファイルに追加するようにユーザーに指示します。

  ```
  ## sshd_config
  IgnoreUserKnownHosts no
  ```

  ユーザーへの指示については、371 ページの「Secure Shell で使用する公開鍵と非公開鍵のペアを生成する方法」を参照してください。

- クライアントの公開鍵をサーバーにコピーします。

  ホスト鍵は、/etc/ssh ディレクトリに格納されます。鍵は、通常、最初のブート時に sshd デーモンによって生成されます。

  a. サーバー上の **/etc/ssh/ssh_known_hosts** ファイルに鍵を追加します。

  クライアントで、バックスラッシュなしで 1 行に次のコマンドを入力します。

  ```
  # cat /etc/ssh/ssh_host_dsa_key.pub | ssh RemoteHost \
  'cat >> /etc/ssh/ssh_known_hosts && echo "Host key copied"'
  ```

  b. プロンプトが表示されたら、ログインパスワードを入力します。

  ファイルがコピーされると、「Host key copied」というメッセージが表示されます。

  /etc/ssh/ssh_known_hosts ファイルの各行は、スペースで区切られたフィールドで構成されています。

  hostnames algorithm-name publickey comment

  c. **/etc/ssh/ssh_known_hosts** ファイルを編集して、コピーしたエントリの最初のフィールドとして *RemoteHost* を追加します。

  ```
  ## /etc/ssh/ssh_known_hosts File
  RemoteHost  <copied entry>
  ```

例 19–1 ホストに基づく認証を設定する

次の例では、各ホストがサーバーおよびクライアントとして構成されます。一方のホストのユーザーが、他方のホストへの ssh 接続を開始できます。次の構成によって、各ホストがサーバーおよびクライアントになります。

- ホストごとに、 Secure Shell 構成ファイルに次のエントリを入力します。

  ```
  ## /etc/ssh/ssh_config
  HostBasedAuthentication yes
  #
  ## /etc/ssh/sshd_config
  ```

```
HostBasedAuthentication yes
IgnoreRhosts no
```

- ホストごとに、shosts.equiv ファイルに他方のホストに対するエントリを入力します。

```
## /etc/ssh/shosts.equiv on machine2
machine1

## /etc/ssh/shosts.equiv on machine1
machine2
```

- 各ホストの公開鍵を、他方のホストの /etc/ssh/ssh_known_hosts ファイルに入力します。

```
## /etc/ssh/ssh_known_hosts on machine2
... machine1

## /etc/ssh/ssh_known_hosts on machine1
... machine2
```

- ユーザーは、両方のホストにアカウントを持ちます。

```
## /etc/passwd on machine1
jdoe:x:3111:10:J Doe:/home/jdoe:/bin/sh

## /etc/passwd on machine2
jdoe:x:3111:10:J Doe:/home/jdoe:/bin/sh
```

▼ Secure Shell v1 を有効にする方法

この手順は、ホストが v1 および v2 を実行するホストと相互運用されるときに役立ちます。

1 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 両方の **Secure Shell** のプロトコルを使用するホストを構成します。

/etc/ssh/sshd_config ファイルを編集します。

```
# Protocol 2
Protocol 2,1
```

3 **v1** のホスト鍵用に別ファイルを指定します。

HostKey エントリを /etc/ssh/sshd_config ファイルに追加します。

```
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_rsa1_key
```

4 **v1** のホスト鍵を生成します。

```
# ssh-keygen -t rsa1 -f /etc/ssh/ssh_host_rsa1_key -N ''
```

`-t rsa1` v1 の RSA アルゴリズムを示します

`-f` ホスト鍵を保持するファイルを示します。

`-N ''` パスフレーズが必要ないことを示します。

5 **sshd** デーモンを再起動します。

```
# svcadm restart network/ssh:default
```

システムをリブートしても構いません。

▼ Secure Shell のポート転送を構成する方法

ポート転送によって、ローカルポートを遠隔ホストに転送することができるようになります。指定すると、ソケットはローカル側で、そのポートを待機します。また、遠隔側のポートを指定することもできます。

注 – Secure Shell のポート転送では TCP 接続を使用する必要があります。Secure Shell はポート転送のための UDP 接続をサポートしていません。

---

1 **Primary Administrator** 役割を引き受けるか、スーパーユーザーになります。

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 ポート転送ができるように遠隔サーバーで **Secure Shell** の設定を構成します。

`/etc/ssh/sshd_config` ファイルで `AllowTcpForwarding` の値を `yes` に変更します。

```
# Port forwarding
AllowTcpForwarding yes
```

3 **Secure Shell** サービスを再起動します。

```
remoteHost# svcadm restart network/ssh:default
```

永続的なサービスの管理方法については、『Solaris のシステム管理 (基本編)』の第 18 章「サービスの管理 (概要)」および `svcadm`(1M) のマニュアルページを参照してください。

4 ポート転送が使用できることを確認します。

```
remoteHost# /usr/bin/pgrep -lf sshd
 1296 ssh -L 2001:remoteHost:23 remoteHost
```

Oracle Solaris Secure Shell の使用 (作業マップ)

次の作業マップでは、ユーザーが Secure Shell を使用する際の手順を示します。

作業	説明	参照先
公開鍵と非公開鍵のペアを作成します	公開鍵認証を必要とするサイトの Secure Shell にアクセスできるようにします。	371 ページの「Secure Shell で使用する公開鍵と非公開鍵のペアを生成する方法」
パスフレーズを変更します	非公開鍵を認証するフレーズを変更します。	374 ページの「Secure Shell の公開鍵のパスフレーズを変更する方法」
Secure Shell を使用してログインします	遠隔ログイン時に、暗号化された Secure Shell 通信を行うことができるようにします。この方法は、 `rsh` コマンドを使用する場合と同様です。	374 ページの「Secure Shell を使用して遠隔ホストにログインする方法」
パスワードのプロンプトを表示せずに Secure Shell にログインします	パスワードを Secure Shell に提供するエージェントを使用してログインできるようにします。	375 ページの「Secure Shell でのパスワードのプロンプトを減らす方法」
		377 ページの「CDE で `ssh-agent` コマンドが自動的に動作するように設定する方法」
Secure Shell のポート転送を使用します	TCP 経由の Secure Shell 接続で使用するローカルポートまたは遠隔ポートを指定します。	378 ページの「Secure Shell のポート転送を使用する方法」
Secure Shell を使用してファイルをコピーします	ホスト間で安全にファイルをコピーします。	379 ページの「Secure Shell を使用してファイルをコピーする方法」
ファイアウォールの内部のホストから外部のホストに安全に接続します	HTTP または SOCKS5 と互換性のある Secure Shell のコマンドを使用して、ファイアウォールで分断されているホスト間を接続します。	380 ページの「ファイアウォール外部のホストにデフォルト接続を設定する方法」

Oracle Solaris Secure Shell の使用 (手順)

Secure Shell によって、ローカルシェルと遠隔シェルの間の安全なアクセスが可能になります。詳細は、`ssh_config(4)` および `ssh(1)` のマニュアルページを参照してください。

▼ Secure Shell で使用する公開鍵と非公開鍵のペアを生成する方法

使用するサイトがホストに基づく認証またはユーザーの公開鍵認証を実装しているときは、ユーザーは公開鍵と非公開鍵のペアを生成する必要があります。追加のオプションについては、`ssh-keygen(1)` のマニュアルページを参照してください。

始める前に　ホストに基づく認証が構成されているかどうかをシステム管理者に確認します。

1 鍵の生成プログラムを起動します。

```
myLocalHost% ssh-keygen -t rsa
Generating public/private rsa key pair.
...
```

-t はアルゴリズムの種類で、rsa、 dsa、 rsa1 のいずれかです。

2 鍵が格納されるファイルのパスを指定します。

デフォルトでは、RSA v2 の鍵を表すファイル名 id_rsa がカッコ内に表示されます。このファイルを選択するときは、Return キーを押します。代わりのファイル名を入力することもできます。

```
Enter file in which to save the key (/home/jdoe/.ssh/id_rsa):     <Press Return>
```

文字列 .pub を非公開鍵のファイル名に追加すると、自動的に公開鍵のファイル名になります。

3 鍵に使用するパスフレーズを入力します。

このパスフレーズは、非公開鍵を暗号化するときに使用されます。空文字列入力は極力避けてください。入力したパスフレーズは表示されません。

```
Enter passphrase (empty for no passphrase):      <Type passphrase>
```

4 確認のためにパスフレーズを再入力します。

```
Enter same passphrase again:      <Type passphrase>
Your identification has been saved in /home/jdoe/.ssh/id_rsa.
Your public key has been saved in /home/jdoe/.ssh/id_rsa.pub.
The key fingerprint is:
0e:fb:3d:57:71:73:bf:58:b8:eb:f3:a3:aa:df:e0:d1 jdoe@myLocalHost
```

5 結果を確認します。

鍵ファイルへのパスが正しいことを確認します。

```
% ls ~/.ssh
id_rsa
id_rsa.pub
```

この時点で公開鍵と非公開鍵のペアが作成されました。

6 適切なオプションを選択します。

- 管理者がホストに基づく認証を構成しているときは、ローカルホストの公開鍵を遠隔ホストにコピーする必要がある場合があります。
  遠隔ホストにログインできるようになっています。詳細については、374 ページの「Secure Shell を使用して遠隔ホストにログインする方法」を参照してください。

  a. 次のコマンドを入力します (ただし、バックスラッシュなしで 1 行に入力)。

  ```
  % cat /etc/ssh/ssh_host_dsa_key.pub | ssh RemoteHost \
  'cat >> ~./ssh/known_hosts && echo "Host key copied"'
  ```

  b. プロンプトが表示されたら、ログインパスワードを入力します。

  ```
  Enter password:      <Type password>
  Host key copied
  %
  ```

- 使用するサイトで公開鍵によるユーザー認証が使用されているときは、遠隔ホストの **authorized_keys** ファイルに反映します。

  a. 公開鍵を遠隔ホストにコピーします。
  次のコマンドを入力します (ただし、バックスラッシュなしで 1 行に入力)。

  ```
  myLocalHost% cat $HOME/.ssh/id_rsa.pub | ssh myRemoteHost \
  'cat >> .ssh/authorized_keys && echo "Key copied"'
  ```

  b. プロンプトが表示されたら、ログインパスワードを入力します。
  ファイルがコピーされると「Key copied」というメッセージが表示されます。

  ```
  Enter password:      Type login password
  Key copied
  myLocalHost%
  ```

7 (省略可能) パスフレーズのプロンプトを減らします。
手順については、375 ページの「Secure Shell でのパスワードのプロンプトを減らす方法」を参照してください。詳細は、`ssh-agent(1)` および `ssh-add(1)` のマニュアルページを参照してください。

例 19–2 ユーザーに対して v1 RSA 鍵を確立する

次の例では、ユーザーが Secure Shell プロトコルの v1 を実行するホストと接続することができます。v1 ホストによって認証されるようにするために、ユーザーは、v1 鍵を作成後、公開鍵の部分を遠隔ホストにコピーします。

```
myLocalHost% ssh-keygen -t rsa1 -f /home/jdoe/.ssh/identity
Generating public/private rsa key pair.
```

```
...
Enter passphrase (empty for no passphrase):     <Type passphrase>
Enter same passphrase again:     <Type passphrase>
Your identification has been saved in /home/jdoe/.ssh/identity.
Your public key has been saved in /home/jdoe/.ssh/identity.pub.
The key fingerprint is:
...
myLocalHost% ls ~/.ssh
id_rsa
id_rsa.pub
identity
identity.pub
myLocalHost% cat $HOME/.ssh/identity.pub | ssh myRemoteHost \
'cat >> .ssh/authorized_keys && echo "Key copied"'
```

▼ Secure Shell の公開鍵のパスフレーズを変更する方法

次の手順で非公開鍵が変更されることはありません。この手順は、非公開鍵 (パスフレーズ) の認証メカニズムを変更するものです。詳細は、ssh-keygen(1) のマニュアルページを参照してください。

- パスフレーズを変更します。

  ssh-keygen コマンドを-p オプションを指定して入力し、プロンプトに答えます。

  ```
  myLocalHost% ssh-keygen -p
  Enter file which contains the private key (/home/jdoe/.ssh/id_rsa):     <Press Return>
  Enter passphrase (empty for no passphrase):     <Type passphrase>
  Enter same passphrase again:     <Type passphrase>
  ```

  -p は、非公開鍵ファイルのパスフレーズの変更を要求します。

▼ Secure Shell を使用して遠隔ホストにログインする方法

1. **Secure Shell セッションを開始します。**

   ssh コマンドを入力して、遠隔ホストの名前を指定します。

   ```
   myLocalHost% ssh myRemoteHost
   ```

   遠隔ホストの信頼性を尋ねるプロンプトが表示されます。

   ```
   The authenticity of host 'myRemoteHost' can't be established.
   RSA key fingerprint in md5 is: 04:9f:bd:fc:3d:3e:d2:e7:49:fd:6e:18:4f:9c:26
   Are you sure you want to continue connecting(yes/no)?
   ```

このプロンプトは、遠隔ホストに初めて接続する場合には正常なプロンプトです。

2 プロンプトが表示されたら、遠隔ホスト鍵の信頼性を確認します。

- 遠隔ホストの信頼性を確認できない場合は、**no** と入力してシステム管理者に連絡します。

  ```
  Are you sure you want to continue connecting(yes/no)? no
  ```

  システム管理者は、大域の /etc/ssh/ssh_known_hosts ファイルを更新する責任があります。更新された ssh_known_hosts ファイルでは、このようなプロンプトは表示されません。

- 遠隔ホストの信頼性を確認したら、プロンプトに答えて次の手順に進みます。

  ```
  Are you sure you want to continue connecting(yes/no)? yes
  ```

3 **Secure Shell** に対して自分を認証します。

a. プロンプトが表示されたら、自分のパスフレーズを入力します。

```
Enter passphrase for key '/home/jdoe/.ssh/id_rsa':      <Type passphrase>
```

b. プロンプトが表示されたら、アカウントのパスワードを入力します。

```
jdoe@myRemoteHost's password:      <Type password>
Last login: Fri Jul 20 14:24:10 2001 from myLocalHost
myRemoteHost%
```

4 遠隔ホストでトランザクションを実行します。
ユーザーが送信するコマンドはすべて暗号化されます。ユーザーが受信する応答はすべて暗号化されます。

5 **Secure Shell** の接続を切断します。
終了したら、**exit** を入力するか、通常の方法でシェルを終了します。

```
myRemoteHost% exit
myRemoteHost% logout
Connection to myRemoteHost closed
myLocalHost%
```

▼ Secure Shell でのパスワードのプロンプトを減らす方法

Secure Shell の使用に際してパスフレーズやパスワードを入力しない場合は、エージェントデーモンを使用できます。セッションを開始するときにデーモンを起動します。次に、エージェントデーモンを使用して非公開鍵を格納するために、ssh-add コマンドを使用します。ホストごとにアカウントが異なる場合は、セッションに必要な非公開鍵を追加します。

エージェントデーモンの起動は、次の手順で説明するように、必要に応じて手動で行うことができます。各セッションを開始するときに、エージェントデーモンが自動的に動作するように設定することもできます (377 ページの「CDE で ssh-agent コマンドが自動的に動作するように設定する方法」を参照)。

1 エージェントデーモンを起動します。

```
myLocalHost% eval 'ssh-agent'
Agent pid 9892
```

2 エージェントデーモンが起動していることを確認します。

```
myLocalHost% pgrep ssh-agent
9892
```

3 使用する非公開鍵をエージェントデーモンに追加します。

ssh-add コマンドを入力します。

```
myLocalHost% ssh-add
Enter passphrase for /home/jdoe/.ssh/id_rsa:      <Type passphrase>
Identity added: /home/jdoe/.ssh/id_rsa(/home/jdoe/.ssh/id_rsa)
myLocalHost%
```

4 Secure Shell セッションを開始します。

```
myLocalHost% ssh myRemoteHost
```

パスフレーズを要求するプロンプトは表示されません。

例 19–3 ssh-add オプションを使用する

この例では、jdoe が 2 つの鍵をエージェントデーモンに追加します。-l オプションは、デーモンに格納されているすべての鍵を一覧表示するために使用します。セッションの最後に、-D オプションを使用して、エージェントデーモンからすべての鍵を削除します。

```
myLocalHost% ssh-agent
myLocalHost% ssh-add
Enter passphrase for /home/jdoe/.ssh/id_rsa:      <Type passphrase>
Identity added: /home/jdoe/.ssh/id_rsa(/home/jdoe/.ssh/id_rsa)
myLocalHost% ssh-add /home/jdoe/.ssh/id_dsa
Enter passphrase for /home/jdoe/.ssh/id_dsa:      <Type passphrase>
Identity added:
/home/jdoe/.ssh/id_dsa(/home/jdoe/.ssh/id_dsa)

myLocalHost% ssh-add -l
md5 1024 0e:fb:3d:53:71:77:bf:57:b8:eb:f7:a7:aa:df:e0:d1
/home/jdoe/.ssh/id_rsa(RSA)
md5 1024 c1:d3:21:5e:40:60:c5:73:d8:87:09:3a:fa:5f:32:53
/home/jdoe/.ssh/id_dsa(DSA)
```

User conducts Oracle Solaris Secure Shell transactions

```
myLocalHost% ssh-add -D
Identity removed:
/home/jdoe/.ssh/id_rsa(/home/jdoe/.ssh/id_rsa.pub)
/home/jdoe/.ssh/id_dsa(DSA)
```

▼ CDE で ssh-agent コマンドが自動的に動作するように設定する方法

CDE を使用する場合、Secure Shell を使用するときにパスフレーズとパスワードを入力しないようにするには、エージェントデーモン `ssh-agent` を自動的に起動します。このエージェントデーモンは、`.dtprofile` スクリプトから起動できます。パスフレーズとパスワードをエージェントデーモンに追加する方法については、例 19–3 を参照してください。

注意 – Sun Java Desktop System (Java DS) を使用する場合には、`ssh-agent` コマンドが自動的に実行されるように設定しないでください。`ssh-agent` プロセスの終了は CDE インタフェースによって制御されるため、Java DS を終了してもデーモンは動作し続けます。たとえば、CDE セッションでデーモンを起動し、Java DS セッションに移動してからログアウトした場合、デーモンは動作し続けます。

実行中のデーモンはシステムリソースを使用します。`ssh-agent` デーモンを実行したまま放置しても、何らかの既知の問題が発生するわけではありませんが、このデーモンにはパスワードが含まれているので、セキュリティーが脅かされる恐れがあります。

1. ユーザーの起動スクリプトでエージェントデーモンを自動的に起動します。

   `$HOME/.dtprofile` スクリプトの最後に次の行を追加します。

   ```
   if [ "$SSH_AUTH_SOCK" = "" -a -x /usr/bin/ssh-agent ]; then
                   eval `/usr/bin/ssh-agent`
   fi
   ```

2. CDE セッションを終了するときにエージェントデーモンを終了します。

   `$HOME/.dt/sessions/sessionexit` スクリプトに次の行を追加します。

   ```
   if [ "$SSH_AGENT_PID" != "" -a -x /usr/bin/ssh-agent ]; then
                   /usr/bin/ssh-agent -k
   fi
   ```

   このエントリにより、CDE セッションが終了したあとで、Secure Shell エージェントは使用できなくなります。このスクリプトは CDE に固有のインタフェース `sessionexit` を使用しているため、Sun Java Desktop System セッションでこの手順を実行しても、エージェントデーモンは終了しません。

▼ Secure Shell のポート転送を使用する方法

遠隔ホストに転送されるローカルポートを指定することができます。指定すると、ソケットはローカル側で、そのポートを待機します。このポートから遠隔ホストへの接続は、セキュリティー保護されたチャネルを介して行われます。たとえば、ポート 143 を指定すれば、IMAP4 で電子メールを遠隔で取得できます。また、遠隔側のポートを指定することもできます。

始める前に　ポート転送を使用するために、管理者は、遠隔の Secure Shell サーバーでのポート転送を有効にする必要があります。詳細については、370 ページの「Secure Shell のポート転送を構成する方法」を参照してください。

- ポート転送を安全に使用するために、次のオプションのどちらかを選択してください。
  - ローカルポートを遠隔ポートからのセキュリティー保護された通信の受信側として設定するには、両方のポートを指定します。

    遠隔からの通信を待機するローカルポートを指定します。また、通信を転送する遠隔ホストと遠隔ポートを指定します。

    ```
    myLocalHost% ssh -L localPort:remoteHost:remotePort
    ```

  - 遠隔ポートをローカルポートからのセキュリティー保護された接続の受信側として設定するには、両方のポートを指定します。

    遠隔からの通信を待機する遠隔ポートを指定します。また、通信を転送するローカルホストとローカルポートを指定します。

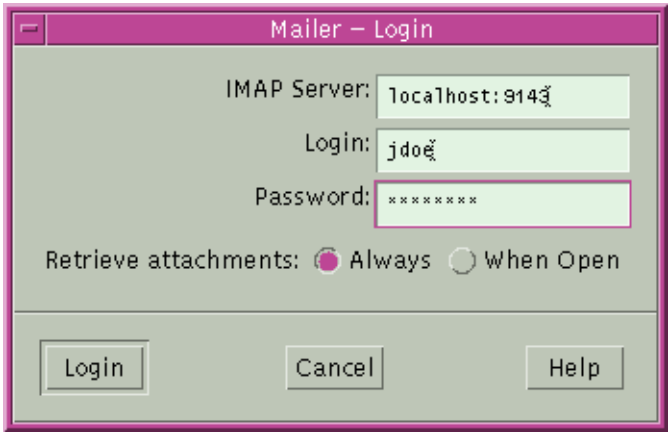
    ```
    myLocalHost% ssh -R remotePort:localhost:localPort
    ```

例 19–4　ローカルポート転送を使用してメールを受信する

次の例は、ローカルポート転送を使用して、遠隔サーバーからのメールを安全に受信する方法を示しています。

```
myLocalHost% ssh -L 9143:myRemoteHost:143 myRemoteHost
```

このコマンドは、myLocalHost のポート9143 からポート 143 に接続を転送します。ポート 143 は、myRemoteHost の IMAP v2 のサーバーポートです。ユーザーがメールアプリケーションを起動するときは、次のダイアログボックスのように、ローカルポート番号を指定する必要があります。



ダイアログボックスの localhost を myLocalHost と混同しないようにしてください。myLocalHost は仮のホスト名です。localhost はローカルシステムを表すキーワードです。

例 19–5 遠隔ポート転送を使用してファイアウォールの外部と通信する

この例では、エンタープライズ環境のユーザーが、外部ネットワーク上のホストから企業のファイアウォール内部のホストに接続を転送する方法を示しています。

```
myLocalHost% ssh -R 9022:myLocalHost:22 myOutsideHost
```

このコマンドは、myOutsideHost 上のポート 9022 からローカルホスト上のポート 22 (sshd サーバー) に接続を転送します。

```
myOutsideHost% ssh -p 9022 localhost
myLocalHost%
```

▼ Secure Shell を使用してファイルをコピーする方法

次の手順では、scp コマンドを使用して、暗号化されたファイルをホスト間でコピーする方法を示します。暗号化されたファイルは、ローカルホストと遠隔ホストとの間、または 2 つの遠隔ホスト間でコピーできます。scp コマンドは、rcp コマンドと同様に動作しますが、認証を要求するプロンプトを表示する点で異なります。詳細は、scp(1) のマニュアルページを参照してください。

ftp コマンドより安全な sftp を使用することもできます。詳細は、sftp(1) のマニュアルページを参照してください。例については、例 19–6 を参照してください。

1 セキュリティー保護されたコピープログラムを起動します。

ソースファイル、遠隔コピー先のユーザー名、およびコピー先ディレクトリを指定します。

```
myLocalHost% scp myfile.1 jdoe@myRemoteHost:~
```

2 プロンプトが表示されたら、パスフレーズを入力します。

```
Enter passphrase for key '/home/jdoe/.ssh/id_rsa':      <Type passphrase>
myfile.1       25% |*******                      |    640 KB  0:20 ETA
myfile.1
```

パスフレーズを入力すると、進行状況インジケータが表示されます。上記の出力の 2 行目が進行状況インジケータです。進行状況インジケータには、次の項目が表示されます。

- ファイル名
- そのファイル全体に対する、転送が完了した量の割合 (%)
- そのファイル全体に対する、転送が完了した量の割合を示すアスタリスク (*)
- 転送が完了したデータの量
- ファイル全体が転送されるまでの推定時間 (ETA)。推定残り時間

例 19–6 sftp コマンドを使用するときにポートを指定する

この例では、ユーザーは sftp コマンドで特定のポートを使用しようとしています。ユーザーは -o オプションを使用してポートを指定します。

```
% sftp -o port=2222 guest@RemoteFileServer
```

▼ ファイアウォール外部のホストにデフォルト接続を設定する方法

Secure Shell を使用して、ファイアウォール内部のホストからファイアウォール外部のホストに接続することができます。接続するには、構成ファイル内またはコマンド行オプションに ssh のプロキシコマンドを指定します。コマンド行オプションについては、例 19–7 を参照してください。

通常は、構成ファイルを使用して、ssh の対話操作をカスタマイズします。

- 1 つの方法として、~/.ssh/config の個人用構成ファイルをカスタマイズします。
- もう 1 つの方法として、管理構成ファイル /etc/ssh/ssh_config を使用します。

ファイルは、2 種類のプロキシコマンドでカスタマイズできます。一方が HTTP 接続用、もう一方が SOCKS5 接続用です。詳細は、ssh_config(4) のマニュアルページを参照してください。

1 構成ファイルにプロキシコマンドとホストを指定します。

次の構文を使用して、必要なプロキシコマンドとホストの数に応じて行を追加します。

```
[Host outside-host]
ProxyCommand proxy-command [-h proxy-server] \
[-p proxy-port] outside-host|%h outside-port|%p
```

`Host` *outside-host*
コマンド行でリモートホスト名を指定した場合、プロキシコマンド指定をインスタンスに限定します。*outside-host* でワイルドカードを使用した場合、一連のホストに対してプロキシコマンド指定が適用されます。

proxy-command
プロキシコマンドを指定します。

次のいずれかを指定できます。

- HTTP 接続の場合は、`/usr/lib/ssh/ssh-http-proxy-connect`
- SOCKS5 接続の場合は、`/usr/lib/ssh/ssh-socks5-proxy-connect`

`-h` *proxy-server* と `-p` *proxy-port*
これらのオプションは、プロキシサーバーとプロキシポートをそれぞれ指定します。これらのオプションは、`HTTPPROXY`、 `HTTPPROXYPORT`、 `SOCKS5_PORT`、`SOCKS5_SERVER`、 `http_proxy` などの、プロキシサーバーとプロキシポートを指定するどのような環境変数よりも優先されます。`http_proxy` 変数は URL を指定します。これらのオプションを指定しない場合、適切な環境変数を設定する必要があります。詳細は、`ssh-socks5-proxy-connect(1)` および `ssh-http-proxy-connect(1)` のマニュアルページを参照してください。

outside-host
接続先のホストを指定します。`%h` 代入引数を使うとコマンド行からホストを指定できます。

outside-port
接続先のポートを指定します。`%p` 代入引数を使うとコマンド行からポートを指定できます。`Host` *outside-host* オプションを使わずに `%h` と `%p` を指定した場合、`ssh` コマンドが呼び出されるたびに、引数に指定されたホストにプロキシコマンドが適用されます。

2 外部のホストを指定して、**Secure Shell** を実行します。

たとえば、次のように入力します。

```
myLocalHost% ssh myOutsideHost
```

このコマンドは、個人用構成ファイル内で `myOutsideHost` のプロキシコマンド指定を検索します。指定が検出されない場合、このコマンドは、システム全体の構成ファイル `/etc/ssh/ssh_config` から検索します。プロキシコマンドが `ssh` コマンドに置き換わります。

例 19–7 コマンド行からファイアウォール外部のホストに接続する

380 ページの「ファイアウォール外部のホストにデフォルト接続を設定する方法」では、構成ファイルでプロキシコマンドを指定する方法について説明しました。この例では、プロキシコマンドを ssh コマンド行で指定します。

```
% ssh -o'Proxycommand=/usr/lib/ssh/ssh-http-proxy-connect \
-h myProxyServer -p 8080 myOutsideHost 22' myOutsideHost
```

ssh コマンドの -o オプションには、プロキシコマンドを指定するコマンド行を入力できます。この例のコマンドは次のことを行います。

- ssh を HTTP プロキシコマンドに置き換える
- プロキシサーバーとして、ポート 8080 および myProxyServer を使用する
- myOutsideHost のポート 22 に接続する

◆ ◆ ◆ 第 20 章

Oracle Solaris Secure Shell (参照)

この章では、Oracle Solaris の Secure Shell 機能の構成オプションについて説明します。この章の内容は次のとおりです。

Secure Shell を構成する手順については、第 19 章「Oracle Solaris Secure Shell の使用 (手順)」を参照してください。

標準的な Secure Shell セッション

Secure Shell デーモン (`sshd`) は通常、ネットワークサービスが開始されるブート時に起動されます。デーモンは、クライアントからの接続を待機します。Secure Shell セッションは、`ssh`、`scp`、または `sftp` コマンドが実行されると開始します。接続を受信するたびに、新しい `sshd` デーモンがフォークされます。フォークされたデーモンは、鍵の交換、暗号化、認証、コマンドの実行、およびクライアントとのデータ交換を行います。Secure Shell セッションの特性は、クライアント構成ファイルとサーバー構成ファイルによって決定されます。コマンド行引数は、構成ファイルの設定より優先されます。

クライアントとサーバーは、相互に認証する必要があります。認証に成功したあと、ユーザーはコマンドを遠隔で実行でき、ホスト間でデータをコピーできます。

Secure Shell でのセッションの特性

サーバー側の sshd デーモンの動作は、/etc/ssh/sshd_config ファイルのキーワードを設定することで変更できます。たとえば、sshd_config ファイルにより、サーバーにアクセスするときに使用する認証タイプを変更できます。サーバー側の動作は、sshd デーモンを起動するときに、コマンド行オプションによって変更することもできます。

クライアント側の動作は、Secure Shell のキーワードで変更できます。キーワードの優先順位は次のとおりです。

- コマンド行オプション
- ユーザーの構成ファイル (~/.ssh/config)
- システム全体の構成ファイル (/etc/ssh/ssh_config)

たとえば、aes128–ctr を優先しているシステム全体の Ciphers の設定を上書きするには、コマンド行に -c aes256–ctr,aes128-ctr,arcfour のように指定します。これで、最初の暗号化方式 aes256–ctr が優先されるようになります。

Secure Shell での認証と鍵の交換

Secure Shell の v1 のプロトコルと v2 のプロトコルは両方とも、クライアントのユーザーとホストの認証およびサーバーのホスト認証をサポートしています。両プロトコルは、Secure Shell セッションの保護のためにセッション暗号化鍵の交換を必要とします。各プロトコルには、認証と鍵の交換の方式がいくつかあります。その中には、任意のものもあります。Secure Shell は、多数のクライアント認証メカニズムをサポートしています (表 19–1 参照)。サーバーは、既知のホスト公開鍵を使用して認証されます。

v1 のプロトコルでは、Secure Shell は、パスワードによるユーザー認証をサポートしています。v1 のプロトコルは、ユーザー公開鍵と信頼されるホスト公開鍵による認証もサポートしています。サーバー認証は、ホスト公開鍵によって行われます。v1 のプロトコルでは、公開鍵はすべて RSA 鍵です。セッション鍵の交換には、定期的に再生成される短期サーバー鍵の使用が必要です。

v2 のプロトコルでは、Secure Shell は、ユーザー認証と汎用対話型認証をサポートしています。汎用対話型認証は、通常、パスワードを必要とします。v2 のプロトコルは、ユーザー公開鍵および信頼されるホスト公開鍵による認証もサポートしています。鍵は、RSA の場合と DSA の場合があります。セッション鍵の交換は、サーバー認証手順で署名される Diffie-Hellman 短期鍵の交換で構成されます。さらに、Secure Shell は認証に GSS 資格を使用することもできます。

Secure Shell での GSS 資格の取得

Secure Shell で認証のために GSS-API を使用するには、サーバーが GSS-API のアクセプタの資格を持ち、クライアントが GSS-API のイニシエータの資格を持つ必要があります。mech_dh および mech_krb5 がサポートされています。

mech_dh では、サーバーは、root が keylogin コマンドを実行すると、GSS-API アクセプタの資格を持ちます。

mech_krb5 では、サーバーは、サーバーに対応するホスト主体が /etc/krb5/krb5.keytab で有効なエントリを持つと、GSS-API アクセプタの資格を持ちます。

クライアントは、次のどちらかによって、mech_dh に対するイニシエータの資格を持ちます。

- keylogin コマンドが実行された。
- pam_dhkeys モジュールが pam.conf ファイルで使用されている。

クライアントは、次のどちらかによって、mech_krb5 に対するイニシエータの資格を持ちます。

- kinit コマンドが実行された。
- pam_krb5 モジュールが pam.conf ファイルで使用されている。

Secure RPC での mech_dh の使用方法については、第 16 章「認証サービスの使用 (手順)」を参照してください。mech_krb5 の使用方法については、第 21 章「Kerberos サービスについて」を参照してください。メカニズムの詳細については、mech(4) および mech_spnego(5) のマニュアルページを参照してください。

Secure Shell でのコマンドの実行とデータの転送

認証が完了すると、ユーザーは通常、シェルまたはコマンド実行を要求して Secure Shell を使用します。ユーザーは、ssh のオプションを使用して、要求を行うことができます。要求には、擬似端末の配置、X11 または TCP/IP の接続の転送、セキュリティー保護された接続上での ssh-agent 認証プログラムの有効化などがあります。

ユーザーセッションの基本手順は次のとおりです。

1. ユーザーがシェルまたはコマンドの実行を要求し、セッションモードを開始します。

   セッションモードでは、クライアント側では、データは端末を通して送受信されます。また、サーバー側ではシェルまたはコマンドを介して送受信されます。
2. データの転送が完了すると、ユーザープログラムは終了します。
3. 既存の接続を除いて、すべての X11 接続と TCP/IP 接続の転送を停止します。既存の X11 接続と TCP/IP 接続は、開いたままです。

4. サーバーは、終了状態のメッセージをクライアントに送信します。開いたままになっていた転送先のポートなど、すべての接続が切断されると、クライアントはサーバーへの接続を切断します。その後、クライアントが終了します。

Secure Shell でのクライアントとサーバーの構成

Secure Shell セッションの特性は、構成ファイルで変更できます。構成ファイルに対しては、コマンド行オプションを使用することで、ある程度優先することができます。

Secure Shell でのクライアントの構成

ほとんどの場合、Secure Shell セッションのクライアント側の特性は、システム全体の構成ファイル (`/etc/ssh/ssh_config`) によって決定されます。ユーザーの構成ファイル (`~/.ssh/config`) は、`ssh_config` ファイル内の設定より優先されます。さらに、コマンド行での指定は、これらの構成ファイルより優先されます。

サーバーの `/etc/ssh/sshd_config` ファイル内の設定によって、クライアント側のどの要求がサーバーによって許可されるかが決まります。サーバー構成の設定の一覧については、387 ページの「Secure Shell でのキーワード」を参照してください。詳細は、`sshd_config(4)` のマニュアルページを参照してください。

クライアントの構成ファイルのキーワードは、387 ページの「Secure Shell でのキーワード」 に一覧表示されています。キーワードにデフォルト値がある場合は、その値が指定されます。これらのキーワードについては、`ssh(1)`、`scp(1)`、`sftp(1)`、および `ssh_config(4)` のマニュアルページに詳細を記載しています。アルファベット順のキーワードと相当するコマンド行の優先指定の一覧については、表 20–8 を参照してください。

Secure Shell でのサーバーの構成

サーバー側の Secure Shell セッションの特性は、`/etc/ssh/sshd_config` ファイルによって管理されます。サーバーの構成ファイルのキーワードは、387 ページの「Secure Shell でのキーワード」 に一覧表示されています。キーワードにデフォルト値がある場合は、その値が指定されます。キーワードの詳細については、`sshd_config(4)` のマニュアルページを参照してください。

Secure Shell でのキーワード

次の表は、キーワードおよびそのデフォルト値 (存在する場合) の一覧です。キーワードはアルファベット順になっています。クライアント側のキーワードは、ssh_config ファイルにあります。サーバーに適用されるキーワードは、sshd_config ファイルにあります。両方のファイルで設定されているキーワードもあります。キーワードが 1 つのプロトコルのバージョンにのみ適用される場合には、そのバージョンが記載されています。

表 20–1 Secure Shell 構成ファイルでのキーワード (A から Escape まで)

キーワード	デフォルト値	場所	プロトコル
AllowGroups	デフォルトなし	サーバー	
AllowTcpForwarding	yes	サーバー	
AllowUsers	デフォルトなし	サーバー	
AuthorizedKeysFile	~/.ssh/authorized_keys	サーバー	
Banner	/etc/issue	サーバー	
Batchmode	no	クライアント	
BindAddress	デフォルトなし	クライアント	
CheckHostIP	yes	クライアント	
ChrootDirectory	no	サーバー	v2
Cipher	blowfish、 3des	クライアント	v1
Ciphers	aes128-ctr、 aes128-cbc、 3des-cbc、 blowfish-cbc、 arcfour	両方	v2
ClearAllForwardings	no	クライアント	
ClientAliveCountMax	3	サーバー	v2
ClientAliveInterval	0	サーバー	v2
Compression	no	両方	
CompressionLevel	デフォルトなし	クライアント	v1

表 20–1 Secure Shell 構成ファイルでのキーワード (A から Escape まで) (続き)

キーワード	デフォルト値	場所	プロトコル
ConnectionAttempts	1	クライアント	
DenyGroups	デフォルトなし	サーバー	
DenyUsers	デフォルトなし	サーバー	
DynamicForward	デフォルトなし	クライアント	
EscapeChar	~	クライアント	

表 20–2 Secure Shell 構成ファイルでのキーワード (Fall から Local まで)

キーワード	デフォルト値	場所	プロトコル
FallBackToRsh	no	クライアント	
ForwardAgent	no	クライアント	
ForwardX11	no	クライアント	
GatewayPorts	no	両方	
GlobalKnownHostsFile	/etc/ssh/ssh_known_hosts	クライアント	
GSSAPIAuthentication	yes	両方	v2
GSSAPIDelegateCredentials	no	クライアント	v2
GSSAPIKeyExchange	yes	両方	v2
GSSAPIStoreDelegateCredentials	yes	サーバー	v2
Host	* 詳細については、391 ページの「Secure Shell でのホスト固有のパラメータ」を参照してください。	クライアント	
HostbasedAuthentication	no	両方	v2
HostbasedUsesNameFromPacketOnly	no	サーバー	v2
HostKey	/etc/ssh/ssh_host_key	サーバー	v1

表 20–2 Secure Shell 構成ファイルでのキーワード (Fall から Local まで) (続き)

キーワード	デフォルト値	場所	プロトコル
HostKey	/etc/ssh/host_rsa_key、/etc/ssh/host_dsa_key	サーバー	v2
HostKeyAlgorithms	ssh-rsa、 ssh-dss	クライアント	v2
HostKeyAlias	デフォルトなし	クライアント	v2
HostName	デフォルトなし	クライアント	v2
IdentityFile	~/.ssh/identity	クライアント	v1
IdentityFile	~/.ssh/id_dsa、~/.ssh/id_rsa	クライアント	v2
IgnoreRhosts	yes	サーバー	
IgnoreUserKnownHosts	yes	サーバー	
KbdInteractiveAuthentication	yes	両方	
KeepAlive	yes	両方	
KeyRegenerationInterval	3600 (秒)	サーバー	
ListenAddress	デフォルトなし	サーバー	
LocalForward	デフォルトなし	クライアント	

表 20–3 Secure Shell 構成ファイルでのキーワード (Login から R まで)

キーワード	デフォルト値	場所	プロトコル
LoginGraceTime	600 (秒)	サーバー	
LogLevel	info	両方	
LookupClientHostnames	yes	サーバー	
MACs	hmac-sha1、 hmac-md5	両方	v2
MaxAuthTries	6	サーバー	
MaxAuthTriesLog	3	サーバー	
MaxStartups	10:30:60	サーバー	

表 20–3 Secure Shell 構成ファイルでのキーワード (Login から R まで) (続き)

キーワード	デフォルト値	場所	プロトコル
NoHostAuthenticationForLocalHost	no	クライアント	
NumberOfPasswordPrompts	3	クライアント	
PAMAuthenticationViaKBDInt	yes	サーバー	v2
PasswordAuthentication	yes	両方	両方
PermitEmptyPasswords	no	サーバー	
PermitRootLogin	no	サーバー	
PermitUserEnvironment	no	サーバー	
PidFile	/var/run/sshd.pid	サーバー	
Port	22	両方	
PreferredAuthentications	hostbased,publickey,keyboard-interactive,passwor	クライアント	v2
PrintLastLog	yes	サーバー	v2
PrintMotd	no	サーバー	
Protocol	2,1	両方	
ProxyCommand	デフォルトなし	クライアント	
PubkeyAuthentication	yes	両方	v2
RemoteForward	デフォルトなし	クライアント	
RhostsAuthentication	no	両方	v1
RhostsRSAAuthentication	no	両方	v1
RSAAuthentication	no	両方	v1

表 20–4 Secure Shell 構成ファイルでのキーワード (S から X まで)

キーワード	デフォルト値	場所	プロトコル
StrictHostKeyChecking	ask	クライアント	
StrictModes	yes	サーバー	

表 20–4 Secure Shell 構成ファイルでのキーワード (S から X まで) (続き)

キーワード	デフォルト値	場所	プロトコル
Subsystem	sftp /usr/lib/ssh/sftp-server	サーバー	
SyslogFacility	auth	サーバー	
UseLogin	no 非推奨として無視される	サーバー	
UseOpenSSLEngine	yes	両方	v2
UsePrivilegedPort	no	両方	v2
User	デフォルトなし	クライアント	
UserKnownHostsFile	~/.ssh/known_hosts	クライアント	
UseRsh	no	クライアント	
VerifyReverseMapping	no	サーバー	
X11DisplayOffset	10	サーバー	
X11Forwarding	yes	サーバー	
X11UseLocalHost	yes	サーバー	
XAuthLocation	/usr/openwin/bin/xauth	両方	

Secure Shell でのホスト固有のパラメータ

ローカルホストごとに異なる Secure Shell 特性を使用すると便利な場合、システム管理者は適用される /etc/ssh/ssh_config ファイルにホストまたはその正規表現形式に従って別々のパラメータセットを定義できます。ファイル内のエントリを、Host キーワードでグループ化してください。Host キーワードを使用しない場合、クライアント構成ファイル内のエントリは、ユーザーが使用しているローカルホストに適用されます。

Secure Shell およびログインの環境変数

次の Secure Shell キーワードが sshd_config ファイルに存在しないときは、/etc/default/login ファイルの相当するエントリから値を取得します。

/etc/default/login のエントリ	sshd_config のキーワードと値
CONSOLE=*	PermitRootLogin=without-password
#CONSOLE=*	PermitRootLogin=yes
PASSREQ=YES	PermitEmptyPasswords=no
PASSREQ=NO	PermitEmptyPasswords=yes
#PASSREQ	PermitEmptyPasswords=no
TIMEOUT=*secs*	LoginGraceTime=*secs*
#TIMEOUT	LoginGraceTime=300
RETRIES および SYSLOG_FAILED_LOGINS	password および keyboard-interactive 認証方式にのみ適用される

次の変数は、ユーザーのログインシェルから初期化スクリプトで設定され、sshd デーモンによってその値が使用されます。変数が設定されていない場合には、デーモンはデフォルト値を使用します。

TIMEZONE　TZ 環境変数の設定を制御します。設定されていない場合、sshd デーモンの起動時の TZ の値を使用します。

ALTSHELL　SHELL 環境変数の設定を制御します。デフォルトは ALTSHELL=YES で、sshd デーモンはユーザーのシェルの値を使用します。ALTSHELL=NO の場合、SHELL の値は設定されません。

PATH　PATH 環境変数の設定を制御します。 値が設定されていない場合、デフォルトのパスは /usr/bin になります。

SUPATH　root に対する PATH 環境変数の設定を制御します。値が設定されていない場合、デフォルトのパスは /usr/sbin:/usr/bin になります。

詳細は、login(1) および sshd(1M) のマニュアルページを参照してください。

Secure Shell での既知のホストの管理

ホスト間の通信を安全に行うには、各ローカルホストの /etc/ssh/ssh_known_hosts ファイルにサーバーの公開鍵を格納する必要があります。 /etc/ssh/ssh_known_hosts ファイルを更新するときに、スクリプトを使用することもできますが、セキュリティーが大幅に低下するため、使用しないことを強くお勧めします。

/etc/ssh/ssh_known_hosts ファイルを配布するときは、次のようなセキュリティー保護されたメカニズムで行う必要があります。

- Secure Shell、 IPsec、 または Kerberos を使用した ftp などのセキュリティー保護された接続を使用して、既知の信頼できるマシンから配布する
- システムインストール時に配布する

known_hosts ファイルに偽の公開鍵を挿入してアクセス権を取得しようとする侵入者がいる可能性をなくすには、ssh_known_hosts ファイルの既知の信頼できる入手先として、 JumpStart サーバーを使用します。ssh_known_hosts ファイルは、インストール中に配布できます。あとで、scp コマンドを使用するスクリプトを使用して、最新バージョンを取り込むこともできます。この方法は、JumpStart サーバーから得た公開鍵がすでに各ホストに保管されているため安全です。

Secure Shell のパッケージと初期化

Secure Shell は、 Solaris の主要パッケージと次のパッケージに依存します。

- SUNWgss – Generic Security Service (GSS) ソフトウェアを含みます
- SUNWtcpd – TCP ラッパーを含みます
- SUNWopenssl-libraries – OpenSSL ライブラリを含みます
- SUNWzlib – zip 圧縮ライブラリを含みます

次のパッケージによって Secure Shell がインストールされます。

- SUNWsshr – ルート (/) ディレクトリのクライアントファイルおよびユーティリティーを含みます
- SUNWsshdr – ルート (/) ディレクトリのサーバーファイルおよびユーティリティーを含みます
- SUNWsshcu – /usr ディレクトリの共通ソースファイルを含みます
- SUNWsshdu – /usr ディレクトリのサーバーファイルを含みます
- SUNWsshu – /usr ディレクトリのクライアントファイルおよびユーティリティーを含みます

インストール後システムを再起動すると、sshd デーモンが実行されます。デーモンは、そのシステム上でホスト鍵を作成します。sshd デーモンを実行する Oracle Solaris システムが Secure Shell サーバーです。

Secure Shell ファイル

次の表に、重要な Secure Shell ファイルと推奨されるファイルアクセス権を示します。

表 20–5 Secure Shell ファイル

ファイル名	説明	推奨アクセス権と所有者
/etc/ssh/sshd_config	sshd (Secure Shell デーモン) の構成データを含みます。	-rw-r--r-- root
/etc/ssh/ssh_host_key	ホスト非公開鍵 (v1) を含みます。	-rw------- root
/etc/ssh/ssh_host_dsa_key または /etc/ssh/ssh_host_rsa_key	ホスト非公開鍵 (v2) を含みます。	-rw------- root
host-private-key.pub	ホスト公開鍵を含みます。/etc/ssh/ssh_host_rsa_key.pub など。ホスト鍵をローカル known_hosts ファイルにコピーするときに使用します。	-rw-r--r-- root
/var/run/sshd.pid	Secure Shell デーモン sshd のプロセス ID を含みます。複数のデーモンが実行されている場合は、起動された最後のデーモンを含みます。	-rw-r--r-- root
~/.ssh/authorized_keys	ユーザーアカウントへのログインが許可されているユーザーの公開鍵を保持します。	-rw-r--r-- *username*
/etc/ssh/ssh_known_hosts	このクライアントがセキュリティー保護された通信を行うことのできるすべてのホストのホスト公開鍵を含みます。このファイルはシステム管理者が管理します。	-rw-r--r-- root
~/.ssh/known_hosts	このクライアントがセキュリティー保護された通信を行うことのできるすべてのホストのホスト公開鍵を含みます。このファイルは自動的に管理されます。ユーザーが未知のホストに接続すると、遠隔ホスト鍵がファイルに追加されます。	-rw-r--r-- *username*
/etc/default/login	対応する sshd_config パラメータが設定されていないときの、sshd デーモンのデフォルトを指定します。	-r--r--r-- root
/etc/nologin	このファイルが存在する場合、sshd デーモンは root のログインのみを許可します。このファイルの内容は、ログインしようとするユーザーに対して表示されます。	-rw-r--r-- root
~/.rhosts	ホスト名とユーザー名のペアを含みます。ユーザーは、対応するホストにパスワードを使用しないでログインできます。このファイルは、rlogind デーモンおよび rshd デーモンでも使用されます。	-rw-r--r-- *username*

表 20–5 Secure Shell ファイル (続き)

ファイル名	説明	推奨アクセス権と所有者
~/.shosts	ホスト名とユーザー名のペアを含みます。ユーザーは、対応するホストにパスワードを使用しないでログインできます。このファイルは、ほかのユーティリティーでは使用されません。詳細については、sshd(1M) のマニュアルページの「FILES」節を参照してください。	-rw-r--r-- *username*
/etc/hosts.equiv	.rhosts 認証で使用されるホストを含みます。このファイルは、rlogind デーモンおよび rshd デーモンでも使用されます。	-rw-r--r-- root
/etc/ssh/shosts.equiv	ホストに基づく認証で使用されるホストを含みます。このファイルは、ほかのユーティリティーでは使用されません。	-rw-r--r-- root
~/.ssh/environment	ログイン時の初期割り当てを含みます。デフォルトでは、このファイルは読み取られません。このファイルを読み取るには、sshd_config ファイルの PermitUserEnvironment キーワードが yes に設定されている必要があります。	-rw-r--r-- *username*
~/.ssh/rc	ユーザーシェルが起動する前に実行される初期化ルーチンを含みます。初期化ルーチンの例については、sshd(1M) のマニュアルページを参照してください。	-rw-r--r-- *username*
/etc/ssh/sshrc	システム管理者が用意したホスト固有の初期化ルーチンを含みます。	-rw-r--r-- root
/etc/ssh/ssh_config	クライアントシステムでのシステム設定を構成します。	-rw-r--r-- root
~/.ssh/config	ユーザーの設定を構成します。システム設定より優先されます。	-rw-r--r-- *username*

次の表は、キーワードまたはコマンドオプションが優先される Secure Shell ファイルの一覧です。

表 20–6 優先される Secure Shell ファイルの場所

ファイル名	キーワードの優先指定	コマンド行の優先指定
/etc/ssh/ssh_config		ssh -F *config-file* scp -F *config-file*
~/.ssh/config		ssh -F *config-file*

表 20–6 優先される Secure Shell ファイルの場所 (続き)

ファイル名	キーワードの優先指定	コマンド行の優先指定
/etc/ssh/host_rsa_key	HostKey	
/etc/ssh/host_dsa_key		
~/.ssh/identity	IdentityFile	ssh -i *id-file*
~/.ssh/id_dsa,~/.ssh/id_rsa		scp -i *id-file*
~/.ssh/authorized_keys	AuthorizedKeysFile	
/etc/ssh/ssh_known_hosts	GlobalKnownHostsFile	
~/.ssh/known_hosts	UserKnownHostsFile	
	IgnoreUserKnownHosts	

Secure Shell コマンド

次の表は、主要な Secure Shell コマンドの要約です。

表 20–7 Secure Shell でのコマンド

コマンド	説明	マニュアルページ
ssh	ユーザーを遠隔マシンにログインさせ、遠隔マシン上でコマンドを安全に実行します。このコマンドは、Secure Shell での、rlogin コマンドと rsh コマンドに代わるコマンドです。ssh コマンドは、セキュリティー保護されていないネットワークを介して 2 つの信頼できないホスト間でセキュリティー保護された暗号化通信を行うことを可能にします。X11 接続と任意の TCP/IP ポートも、セキュリティー保護されたチャネルを介して転送されます。	ssh(1)
sshd	Secure Shell 用のデーモンです。このデーモンは、クライアントからの接続を待機します。セキュリティー保護されていないネットワークを介して 2 つの信頼できないホスト間でセキュリティー保護された暗号化通信を行うことを可能にします。	sshd(1M)
ssh-add	RSA または DSA ID を認証エージェント ssh-agent に追加します。ID は「鍵」とも呼ばれます。	ssh-add(1)
ssh-agent	公開鍵認証時に使用される非公開鍵を保持します。ssh-agent プログラムは、X セッションまたはログインセッションの開始時に起動します。ほかのすべてのウィンドウおよびプログラムは、ssh-agent プログラムのクライアントとして起動します。環境変数を使用すれば、ユーザーが ssh コマンドを使用してほかのシステムにログインするときに、エージェントを検出して認証に使用することができます。	ssh-agent(1)
ssh-keygen	Secure Shell の認証鍵を生成および管理します。	ssh-keygen(1)

表 20–7 Secure Shell でのコマンド (続き)

コマンド	説明	マニュアルページ
ssh-keyscan	多数の Secure Shell ホストの公開鍵を収集します。ssh_known_hosts ファイルの作成および検証時に役立ちます。	ssh-keyscan(1)
ssh-keysign	ssh コマンドがローカルホスト上のホスト鍵にアクセスするときに使用されます。Secure Shell v2 によるホストに基づく認証中に必要となるデジタル署名を生成します。このコマンドは、ユーザーではなく ssh コマンドによって呼び出されます。	ssh-keysign(1M)
scp	暗号化された ssh トランスポートを介して、ネットワーク上のホスト間でファイルを安全にコピーします。rcp コマンドと異なり、scp コマンドは、パスワード情報が認証に必要な場合、パスワードまたはパスフレーズを要求します。	scp(1)
sftp	ftp コマンドと同様の対話型ファイル転送プログラムです。ftp コマンドと異なり、sftp コマンドは、暗号化された ssh トランスポートを介してすべての操作を実行します。このコマンドは、指定したホスト名に接続してログインし、対話型コマンドモードに入ります。	sftp(1)

次の表は、Secure Shell のキーワードに優先するコマンドオプションの一覧です。キーワードは、ssh_config ファイルおよび sshd_config ファイルで指定します。

表 20–8 Secure Shell のキーワードに相当するコマンド行

キーワード	ssh コマンド行の優先指定	scp コマンド行の優先指定
BatchMode		scp -B
BindAddress	ssh -b *bind-addr*	scp -a *bind-addr*
Cipher	ssh -c *cipher*	scp -c *cipher*
Ciphers	ssh -c *cipher-spec*	scp -c *cipher-spec*
Compression	ssh -C	scp -C
DynamicForward	ssh -D *SOCKS4-port*	
EscapeChar	ssh -e *escape-char*	
ForwardAgent	ssh -A (有効) ssh -a (無効)	
ForwardX11	ssh -X (有効) ssh -x (無効)	
GatewayPorts	ssh -g	
IPv4	ssh -4	scp -4

表 20–8 Secure Shell のキーワードに相当するコマンド行 (続き)

キーワード	**ssh** コマンド行の優先指定	**scp** コマンド行の優先指定
IPv6	ssh -6	scp -6
LocalForward	ssh -L *localport:remotehost:remoteport*	
MACS	ssh -m *mac-spec*	
Port	ssh -p *port*	scp -P *port*
Protocol	ssh -1 (v1 のみ) ssh -2 (v2 のみ)	
RemoteForward	ssh -R *remoteport:localhost:localport*	

パート VI

Kerberos サービス

この節では、次の章に示す Kerberos サービスの構成、管理、および使用方法について説明します。

- 第 21 章「Kerberos サービスについて」
- 第 22 章「Kerberos サービスの計画」
- 第 23 章「Kerberos サービスの構成 (手順)」
- 第 24 章「Kerberos エラーメッセージと障害追跡」
- 第 25 章「Kerberos 主体とポリシーの管理 (手順)」
- 第 26 章「Kerberos アプリケーションの使用 (手順)」
- 第 27 章「Kerberos サービス (参照)」

◆ ◆ ◆ 第 21 章

Kerberos サービスについて

この章では、Kerberos サービスについて説明します。この章の内容は次のとおりです。

Kerberos サービスとは

「*Kerberos* サービス」は、セキュリティー保護されたネットワーク経由のトランザクションを提供するクライアントサーバー型のアーキテクチャーです。Kerberos サービスでは、強力なユーザー認証とともに、整合性とプライバシを提供します。「認証」により、ネットワークトランザクションの送信者と受信者の識別情報が正しいことが保証されます。さらに Kerberos サービスを使用して、送受信するデータの整合性が検証され(「整合性」)、伝送時にデータが暗号化されます(「プライバシ」)。Kerberos サービスを使用して、他のマシンにログインしてコマンドを実行したり、データを交換したりファイルを安全に転送したりできます。Kerberos サービスは承認サービスも提供するため、システム管理者はサービスやマシンへのアクセスを制限できます。また、Kerberos ユーザーは、自分のアカウントに他人がアクセスするのを制限できます。

Kerberos サービスは「シングルサインオン」システムです。つまり、Kerberos サービスからセッションについて一度だけ認証を受ければ、そのセッションでは、それ以後のすべてのトランザクションが自動的に認証されます。いったん Kerberos サービスから認証されたユーザーは、`ftp`、`rsh` などの Kerberos に基づくコマンドを使用したり、NFS ファイルシステム上のデータにアクセスするたびに、自分自身を認証する必要はありません。つまり、これらのサービスを使用するたびに、ネットワークを介してパスワードを送り、傍受される危険を冒す必要がありません。

Oracle Solaris Kerberos サービスは、マサチューセッツ工科大学 (MIT) で開発された Kerberos V5 ネットワーク認証プロトコルに基づいています。そのため、Kerberos V5 製品を使用したことがあるユーザーは、Oracle Solaris バージョンにもすぐ慣れるはずです。Kerberos V5 プロトコルはネットワークセキュリティーの事実上の業界標準であるため、Oracle Solaris バージョンはほかのシステムとの相互運用性に優れています。つまり、Oracle Solaris Kerberos サービスは Kerberos V5 プロトコルを使用するシステムと協調して動作するため、異機種システム混在のネットワークであってもトランザクションのセキュリティーが保護されます。さらに Kerberos サービスでは、複数のドメイン間でも単一のドメイン内でも認証やセキュリティーの機能を使用できます。

Kerberos サービスには、Oracle Solaris アプリケーションを実行するための柔軟性が備わっています。NFS サービス、`telnet`、`ftp` などのネットワークサービスに関して、Kerberos に基づく要求と Kerberos 以外の要求に対応できるようにサービスを構成できます。このため、Kerberos サービスが有効になっていないシステムで動作するアプリケーションも正しく動作します。もちろん、Kerberos に基づくネットワーク要求だけを許可するように Kerberos サービスを設定することもできます。

Kerberos サービスは、Generic Security Service Application Programming Interface (GSS-API) を使用するアプリケーションの使用時に、認証、整合性、およびプライバシのために Kerberos を使用することができるセキュリティーメカニズムを備えています。ただし、ほかのセキュリティーメカニズムが開発されている場合には、アプリケーションで使用されるセキュリティーメカニズムを Kerberos サービスに限定しておく必要はありません。Kerberos サービスは、GSS-API にモジュールとして統合できるように設計されているため、GSS-API を使用するアプリケーションは、必要に応じたセキュリティーメカニズムを使用できます。

Kerberos サービスの動作

この節では Kerberos 認証システムの概要について説明します。詳細については、587 ページの「Kerberos 認証システムの動作方法」を参照してください。

Kerberos セッションが起動されたあとは、ユーザーから見ると Kerberos サービスが意識されることはほとんどありません。`rsh` や `ftp` などのコマンドは、ほぼ変わりなく動作します。Kerberos セッションの初期化には通常、ログインと Kerberos パスワードの入力しか必要ありません。

Kerberos システムは、「チケット」の概念を中心に動作します。チケットは、ユーザー、および NFS サービスなどのサービスを特定する一連の電子情報です。運転免許証が運転する人と免許の種類を表すのと同じように、チケットもユーザーとユーザーのネットワークアクセス権を表します。Kerberos に基づくトランザクションを実行する (ほかのマシンへの遠隔ログインなど) と、「鍵配布センター (KDC)」に対してチケットの要求が透過的に送信されます。KDC はデータベースにアクセスしてそのユーザーを認証し、そのマシンへのアクセスを許可する

チケットを返します。「透過的」とは、チケットを明示的に要求する必要がないという意味です。この要求は rlogin コマンドの中で行われます。特定のサービスのチケットを取得できるのは認証されたクライアントだけで、別のクライアントが識別情報を仮定して rlogin を使用することはできません。

チケットには一定の属性が与えられています。たとえば、チケットには、新しい認証処理を行わなくても別のマシンで使用できる「転送可能」の属性があります。また、指定の日付まで有効にならない「遅延」の属性もあります。どのユーザーがどの種類のチケットを取得できるかを指定するなど、チケットをどのように使用するかは、「ポリシー」によって設定されます。ポリシーは、Kerberos サービスのインストールや管理の際に決定します。

注 – 「資格」と「チケット」という用語は、頻繁に使用されます。広い意味の Kerberos では、これらの用語は同じ意味で使われることがありますが、技術的には資格は、チケットとそのセッションに対する「セッション鍵」からなります。この違いについては、587 ページの「Kerberos によるサービスへのアクセス」で詳しく説明します。

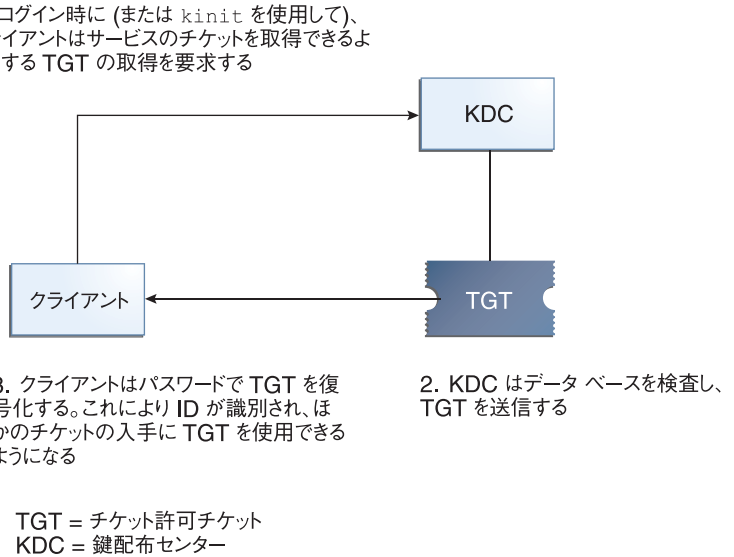
次の節では、Kerberos 認証プロセスについて詳細に説明します。

初期認証: チケット認可チケット (TGT)

Kerberos 認証には、後続の認証を準備する初期認証と、後続の認証の 2 つのフェーズがあります。

次の図では、初期認証の手順を示します。

図 21–1　Kerberos セッションの初期認証



1. クライアント (ユーザー、または NFS などのサービス) は、KDC に TGT を要求して Kerberos セッションを開始します。ほとんどの場合、この要求はログイン時に自動的に実行されます。

   TGT は、ほかの特定のサービスのチケットを取得するために必要です。TGT は、パスポートに似ています。パスポートと同様に、TGT はユーザーを識別して、さまざまなビザの取得をユーザーに許可します。ここでいうビザ (チケット) は、外国に入国するためのものではなく、遠隔マシンやネットワークサービスにアクセスするためのものです。パスポートやビザと同様に、TGT などのチケットには有効期限があります。ただし、Kerberos コマンドは、ユーザーがパスポートを所有していることを通知し、ユーザーに代わってビザを取得します。ユーザー自身がトランザクションを実行する必要はありません。

   チケット認可チケットに類似した例として、4 つのスキー場で使える 3 日間のスキーパスを挙げます。ユーザーは、パスが期限切れになるまで、このパスを任意のスキー場で提示して、そのスキー場のリフトチケットを受け取ります。リフトチケットを入手したら、そのスキー場で好きなだけスキーをすることができます。翌日別のスキー場に行った場合は、またパスを提示して、そのスキー場のリフトチケットを入手します。ただし、Kerberos に基づくコマンドは、ユーザーが週末スキーパスを所有していることをユーザーに通知し、ユーザーに代わってリフトチケットを入手します。したがって、ユーザー自身がトランザクションを実行する必要はありません。

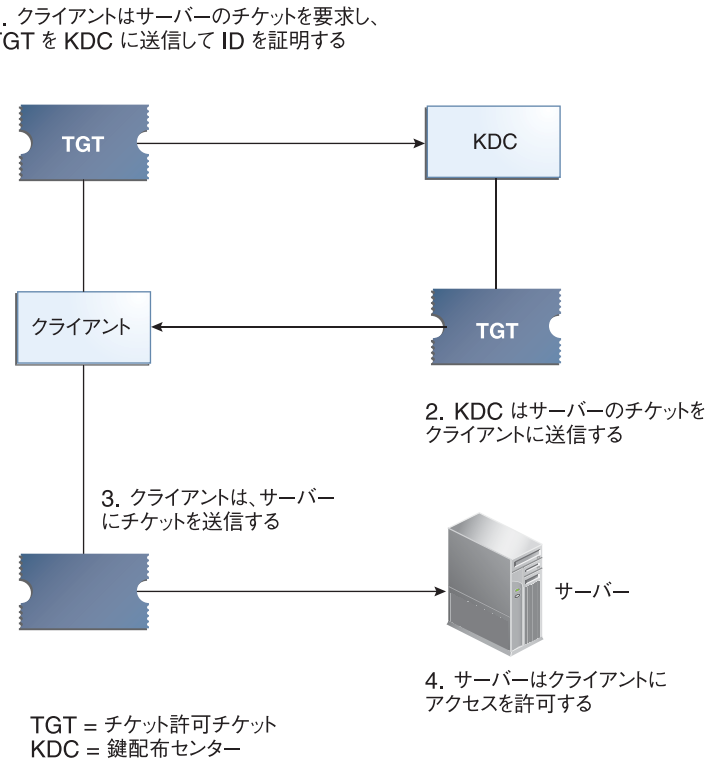
2. KDC は TGT を作成し、それを暗号化してクライアントに送信します。クライアントは、自身のパスワードを使用して TGT を復号化します。

3. クライアントは、有効な TGT を入手したので、TGT が期限切れになるまで、`rlogin`、 `telnet` などあらゆる種類のネットワーク操作チケットを要求できます。この TGT の有効期限は通常、数時間です。クライアントは一意のネットワーク操作を実行するたびに、TGT は KDC にその操作のチケットを要求します。

初期認証後の Kerberos 認証

クライアントが初期認証を受け取ると、後続の認証はそれぞれ次の図のように実行されます。

図 21–2 Kerberos 認証を使用してサービスへのアクセスを取得する



1. クライアントは、別のマシンに遠隔ログインするなど、特定のサービスのチケットを KDC に要求するために、識別情報の証拠として自身の TGT を KDC に送信します。
2. KDC は、そのサービスのチケットをクライアントに送信します。

たとえば、ユーザー joe が、krb5 認証を要する共有を行っている NFS ファイルシステムにアクセスするとします。このユーザーはすでに認証されている (すでに TGT を持っている) ため、そのファイルにアクセスを試みると、NFS クライアントシステムは NFS サービスのチケットを KDC から自動的および透過的に取得します。

たとえば、ユーザー joe がサーバー boston 上で rlogin を使用するとします。このユーザーはすでに認証されている (つまり、すでにチケット認可チケットを持っている) ため、rlogin コマンドの一部として自動的かつ透過的にチケットを取得します。このチケットが期限切れになるまで、このユーザーは必要に応じて boston に遠隔ログインできます。joe がマシン denver に遠隔ログインする場合は、手順 1 の方法で別のチケットを取得します。

3. クライアントはサーバーにチケットを送信します。

   NFS サービスを使用している場合、NFS クライアントは自動的および透過的に NFS サービスのチケットを NFS サーバーに送信します。

4. サーバーはクライアントにアクセス権を許可します。

これらの手順では、サーバーと KDC 間の通信は発生していないように見えます。しかし、サーバーは KDC と通信していて、最初のクライアントと同様に、KDC に自身を登録しています。わかりやすくするために、その部分は省略しています。

Kerberos 遠隔アプリケーション

joe などのユーザーは、次の Kerberos に基づく (Kerberos 化された) コマンドを使用できます。

- ftp
- rcp
- rdist
- rlogin
- rsh
- ssh
- telnet

これらのアプリケーションは、同じ名前の Solaris アプリケーションと同じです。ただし、トランザクションを認証するときに Kerberos 主体を使用できるようにアプリケーションを拡張することにより、Kerberos に基づくセキュリティーを提供します。主体の詳細については、407 ページの「Kerberos 主体」を参照してください。

これらのコマンドについては、568 ページの「Kerberos ユーザーコマンド」で詳しく説明します。

Kerberos 主体

Kerberos サービス内のクライアントは、その「主体 (プリンシパル)」で識別されます。主体は、KDC がチケットを割り当てることができる一意の ID です。主体には、joe などのユーザー、または nfs、 telnet などのサービスがあります。

主体名は慣習により「一次」、「インスタンス」、「レルム」という 3 つの部分からなります。joe/admin@ENG.EXAMPLE.COM は一般的な Kerberos 主体の例です。上記の例では、

- joe が一次です。一次には、この例のようなユーザー名や nfs などのサービスを指定します。また、host を指定することもできます。host を指定すると、ftp、rcp、 rlogin などのさまざまなネットワークサービスを提供する、サービス主体として設定されます。
- admin はインスタンスです。インスタンスは、ユーザー主体の場合はオプションですが、サービス主体では必須です。たとえば、ユーザー joe が必要に応じてシステム管理者の権限を使用する場合は、joe/admin と通常のユーザー ID を使い分けることができます。同じように、joe が 2 つのホストにアカウントを持っている場合、joe/denver.example.com と joe/boston.example.com など、異なるインスタンスで 2 つの主体名を使用することができます。Kerberos サービスでは、joe と joe/admin はまったく別の主体として扱われます。

  サービス主体では、インスタンスは完全指定されたホスト名です。bigmachine.eng.example.com はこのようなインスタンスの例です。この例の一次とインスタンスは、ftp/bigmachine.eng.example.com や host/bigmachine.eng.example.com と表します。
- ENG.EXAMPLE.COM は Kerberos レルムです。レルムについては、407 ページの「Kerberos レルム」を参照してください。

次に有効な主体名を示します。

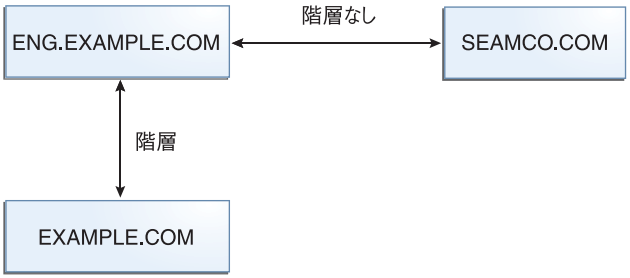
- joe
- joe/admin
- joe/admin@ENG.EXAMPLE.COM
- nfs/host.eng.example.com@ENG.EXAMPLE.COM
- host/eng.example.com@ENG.EXAMPLE.COM

Kerberos レルム

「レルム」とはドメインのようなもので、同じ「マスター KDC」の下にあるシステムをグループとして定義する論理ネットワークです。図 21–3 は、レルム間の関係を示しています。階層構造のレルムでは、1 つのレルムがほかのレルムの上位集合になります。階層ではない (直接接続の) レルムでは、2 つのレルム間のマッピングを定義する必要があります。Kerberos サービスでは、レルム間で共通の認証が可能で

す。その場合、各レルムの KDC に、他のレルムの主体エントリだけが必要になります。Kerberos のこの機能は、「レルム間認証」と呼ばれます。

図 21–3 Kerberos レルム



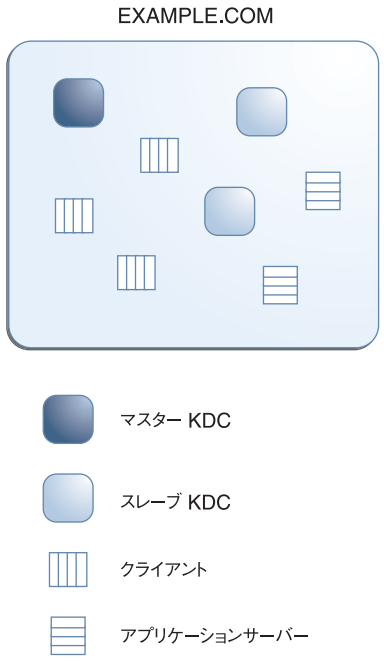
Kerberos サーバー

各レルムには、主体データベースのマスターコピーを保守するサーバーが含まれる必要があります。このサーバーを「マスター KDC サーバー」と呼びます。また各レルムには、主体データベースの重複コピーを保持する「スレーブ KDC サーバー」が少なくとも 1 つ必要です。マスター KDC サーバーおよびスレーブ KDC サーバーは、認証の確立に使用されるチケットを作成します。

レルムにはまた、Kerberos アプリケーションサーバー も含めることができます。このサーバーは、Kerberos サービス (`ftp`、 `telnet`、 `rsh`、 NFS など) へのアクセスを提供します。SEAM 1.0 または 1.0.1 がインストールされている場合、レルムに Kerberos ネットワークアプリケーションサーバーが含まれている可能性がありますが、このソフトウェアはこれらのリリースには含まれていませんでした。

次の図では、レルムの構成例を示します。

図 21–4　一般的な Kerberos レルム



Kerberos セキュリティーサービス

Kerberos サービスは、ユーザーの認証を行うほかに、次の 2 つのセキュリティーサービスを提供します。

- 「整合性」 – 認証が、あるネットワーク上のクライアントが本人であるかどうかを確認するのと同様に、整合性は、クライアントの送信データが有効で、伝送の間に改ざんされていないことを確認します。整合性の確認は、データの暗号チェックサムによって行われます。整合性にはユーザー認証も含まれます。
- 「プライバシ」 – プライバシによって、セキュリティーがさらに向上します。プライバシは、伝送データの整合性を検証するだけでなく、伝送前にデータを暗号化して盗聴を防ぎます。プライバシにもユーザー認証が含まれます。

開発者は、RPCSEC_GSS プログラミングインタフェースを使用することにより、セキュリティーサービスを選択可能な RPC ベースのアプリケーションを設計できます。

複数の Kerberos リリースの構成要素

Kerberos サービスの構成要素は、多数のリリースに組み込まれています。当初、Kerberos サービスと、Kerberos サービスをサポートするベースオペレーティングシステムへの変更は、“Sun Enterprise Authentication Mechanism” (SEAM) という製品名でリリースされました。SEAM 製品の大部分が Oracle Solaris ソフトウェアに組み込まれるようになると、SEAM リリースの内容は減りました。Oracle Solaris リリースで、SEAM 製品のすべての構成要素が組み込まれたので、SEAM 製品の必要性はなくなりました。SEAM という製品名は、歴史的な理由で文書中に存在しています。

次の表は、各リリースに組み込まれている構成要素の一覧です。それぞれの製品リリースは、時系列で示しています。次の節では、すべての構成要素について説明します。

表 21–1 Kerberos リリースの内容

リリース名	内容
Solaris Easy Access Server 3.0 の SEAM 1.0	Solaris 2.6 および Solaris 7 用の Kerberos サービスの完全リリース
Solaris 8 の Kerberos サービス	Kerberos クライアントソフトウェアのみ
Solaris 8 Admin Pack の SEAM 1.0.1	Solaris 8 用の Kerberos KDC とリモートアプリケーション
Solaris 9 の Kerberos サービス	Kerberos KDC とクライアントソフトウェアのみ
SEAM 1.0.2	Solaris 9 用の Kerberos リモートアプリケーション
Solaris 10 の Kerberos サービス	機能が拡張された Kerberos サービスの完全リリース

Kerberos の構成要素

MIT から提供される Kerberos V5 製品と同様に、Oracle Solaris Kerberos サービスには次の構成要素が含まれます。

- 鍵配布センター (KDC):
  - Kerberos データベース管理デーモン – `kadmind`。
  - Kerberos チケット処理デーモン – `krb5kdc`。
  - データベース管理プログラム – `kadmin` (マスターのみ)、 `kadmin.local`、 および `kdb5_util`。
  - データベース伝播ソフトウェア – `kprop` (スレーブのみ) および `kpropd`。
- 資格を管理するためのユーザープログラム – `kinit`、 `klist`、 および `kdestroy`。
- Kerberos パスワードを変更するユーザープログラム – `kpasswd`。

- 遠隔アプリケーション – `ftp`、 `rcp`、 `rdist`、 `rlogin`、 `rsh`、 `ssh`、 および `telnet`。
- 遠隔アプリケーションデーモン – `ftpd`、 `rlogind`、 `rshd`、 `sshd`、 および `telnetd`。
- キータブ管理ユーティリティー – `ktutil`。
- Generic Security Service Application Programming Interface (GSS-API) – アプリケーションは、この API を利用して、複数のセキュリティーメカニズムを使用できます。新しいメカニズムを追加するたびに、アプリケーションをコンパイルし直す必要がありません。GSS-API では、アプリケーションを多くのオペレーティングシステムに移植可能にできる標準インタフェースが使用されています。GSS-API を使用すると、認証サービスだけでなく、整合性およびプライバシセキュリティーサービスをアプリケーションに組み込むことができます。`ftp` と `ssh` は、どちらも GSS-API を使用しています。
- RPCSEC_GSS Application Programming Interface (API) – NFS サービスが Kerberos 認証を使用することができます。RPCSEC_GSS は、使用しているメカニズムに依存しないセキュリティーサービスを提供するセキュリティー様式です。RPCSEC_GSS は、GSS-API 層の最上位に位置しています。GSS_API ベースのセキュリティーメカニズムは、プラグイン可能なので、RPCSEC_GSS を使用するアプリケーションで使用できます。

さらに、Oracle Solaris Kerberos サービスには次の構成要素が含まれています。

- Kerberos グラフィカル管理ツール (`gkadmin`) – 主体および主体ポリシーを管理することができます。この Java テクノロジベースの GUI は、`kadmin` コマンドに代わる機能です。
- PAM 用の Kerberos V5 サービスモジュール – Kerberos サービスのための認証、アカウント管理、セッション管理、およびパスワード管理を提供します。このモジュールを使用すると、Kerberos 認証をユーザーが意識しなくても済むようになります。
- カーネルモジュール – NFS サービスで使用する kerberos サービスのカーネルベースの実装を提供します。これにより、パフォーマンスが大幅に向上します。

Solaris 10 5/08 リリースでの Kerberos の追加機能

Solaris 10 5/08 リリースから、次のような拡張機能が使用できます。

- Solaris Kerberos ソフトウェアが MIT 1.4 バージョンと同期化されました。具体的には、KDC のソフトウェアである `kinit` コマンドと Kerberos メカニズムが更新されました。
- ディレクトリサーバーから LDAP を使用して Kerberos 主体とポリシーのレコードにアクセスする機能のサポートが追加されました。この変更により管理が簡略化されるため、KDC と DS の配備によっては可用性が向上する場合があります。LDAP 関連の手順の一覧については、496 ページの「LDAP ディレクトリサーバーでの KDC の管理」を参照してください。

- このリリースには、追加の設定が不要な Solaris クライアントのサポートが追加されました。Kerberos サービスと一部のデフォルト設定が変更されました。適切に構成された環境では、Solaris Kerberos クライアントがクライアント側の構成なしで機能します。詳細は、425 ページの「クライアントの構成オプション」を参照してください。

Solaris 10 8/07 リリースでの Kerberos の追加機能

Solaris 10 8/07 リリースでは、MIT Kerberos V5 アプリケーションプログラミングインタフェース (krb5-api) がサポートされています。詳細は、`libkrb5`(3LIB) および `krb5-config`(1) のマニュアルページを参照してください。また、mit.edu にある MIT Kerberos V5 プロジェクトの Web ページも参照し、詳細ドキュメントが利用可能になった時点でそれらを入手してください。

krb5-api が使用可能になりましたが、GSS-API は独立したセキュリティーメカニズムであり、IETF 標準でもあるため、GSS-API を使ってネットワークの認証、完全性、プライバシを実現することを強くお勧めします。詳細は、`libgss(3LIB)` のマニュアルページを参照してください。

Solaris 10 6/06 リリースでの Kerberos の追加機能

Solaris 10 6/06 リリースから、`ktkt_warnd` デーモンは、資格の有効期限が近づいてきたとき、ユーザーに警告するだけでなく、資格を自動的に更新するようになりました。資格を自動的に更新するには、ユーザーがログインしている必要があります。

Solaris 10 3/05 リリースでの Kerberos の拡張機能

Oracle Solaris リリースに含まれる Kerberos の機能拡張は、次のとおりです。そのうちのいくつかは、以前の Software Express リリースで採用され、Solaris 10 Beta リリースで更新されたものです。

- Kerberos プロトコルを、`ftp`、 `rcp`、 `rlogin`、 `rsh`、 `ssh`、 および `telnet` などの遠隔アプリケーションでサポートします。詳細は、各コマンドまたはデーモンのマニュアルページおよび `krb5_auth_rules(5)` のマニュアルページを参照してください。
- Kerberos 主体データベースが、毎回データベース全体を転送するのではなく、増分更新によって転送されます。増分更新には、次のような利点があります。
  - サーバー間でのデータベースの整合性が増す
  - 必要なリソース (ネットワーク、CPU など) が少なくてすむ
  - 更新をよりタイムリーに伝播させることができる
  - 伝播を自動化することができる

- Kerberos クライアントの自動構成に役立つ新しいスクリプトが使用できます。このスクリプトは、管理者が Kerberos クライアントを迅速かつ容易に設定するのを支援します。新しいスクリプトの使用手順については、460 ページの「Kerberos クライアントの構成」を参照してください。また、詳細については `kclient(1M)` のマニュアルページを参照してください。
- いくつかの新しい暗号化タイプが Kerberos サービスに追加されました。これらの新しい暗号化タイプによって、セキュリティーが向上し、それらの暗号化タイプをサポートするほかの Kerberos 実装との互換性が強化されます。詳細は、590 ページの「Kerberos 暗号化タイプの使用」を参照してください。追加された暗号化タイプは次のとおりです。
  - AES 暗号化タイプは、Kerberos セッションの高速かつ高セキュリティーの暗号化に使用されます。
  - ARCFOUR-HMAC は、ほかの Kerberos 実装との互換性を強化します。
  - SHA1 での Triple DES (3DES) は、セキュリティーを向上させます。この暗号化タイプにより、この暗号化タイプをサポートする他の Kerberos 実装との相互運用性の強化も図れます。
- 暗号化タイプは、暗号化フレームワークを介して有効になります。このフレームワークは、Kerberos サービスにハードウェアによって高速化された暗号化を提供できます。
- KDC ソフトウェア、ユーザーコマンド、およびユーザーアプリケーションが、TCP ネットワークプロトコルの使用をサポートします。これにより、動作が堅牢になり、Microsoft の Active Directory など、ほかの Kerberos 実装と相互運用性が強化されます。KDC は、従来の UDP ポートと TCP ポートの両方で待機し、どちらのプロトコルを使用する要求にも応答できます。ユーザーコマンドとユーザーアプリケーションは、要求が KDC に送られると、まず UDP を試し、失敗した場合は TCP を試します。
- `kinit` コマンド、`klist` コマンド、および `kprop` コマンドなど、KDC ソフトウェアで IPv6 がサポートされます。IPv6 アドレスがデフォルトでサポートされます。IPv6 のサポートを有効にするために変更する構成パラメータはありません。IPv6 は、`kadmin` コマンドおよび `kadmind` コマンドではサポートされません。
- 新しい `-e` オプションが `kadmin` コマンドのいくつかのサブコマンドに含まれました。この新しいオプションにより、主体の作成中に暗号化タイプを選択できます。詳細は、`kadmin(1M)` のマニュアルページを参照してください。
- `pam_krb5` モジュールに、PAM フレームワークを使って Kerberos 資格キャッシュを管理する機能が追加されました。詳細は、`pam_krb5(5)` のマニュアルページを参照してください。
- DNS 検索を使用することにより、Kerberos KDC、 admin サーバー、kpasswd サーバー、およびホスト名またはドメイン名のレルムへのマッピングの自動検出がサポートされます。この拡張機能により、Kerberos クライアントのインストールに必要な手順のいくつかが必要なくなります。Kerberos クライアント

は、構成ファイルを読み取る代わりに DNS を使用して、KDC サーバーを検出することができます。詳細は、krb5.conf(4) のマニュアルページを参照してください。

- pam_krb5_migrate と呼ばれる新しい PAM モジュールが追加されました。この新しいモジュールは、ユーザーが Kerberos アカウントを持たない場合にローカルの Kerberos レルムに自動的に移行するのを支援します。詳細は、pam_krb5_migrate(5) のマニュアルページを参照してください。
- ~/.k5login ファイルが GSS アプリケーションの ftp および ssh で使用できます。詳細は、gss_auth_rules(5) のマニュアルページを参照してください。
- kproplog ユーティリティーが更新され、ログエントリごとにすべての属性名を出力できます。詳細は、kproplog(1M) のマニュアルページを参照してください。
- krb5.conf ファイルの構成オプションを使用して、厳格な TGT 検証機能を無効にできるようになりました。詳細は、krb5.conf(4) のマニュアルページを参照してください。
- パスワード変更ユーティリティーが拡張され、Oracle Solaris Kerberos V5 管理サーバーが、Oracle Solaris ソフトウェアを実行していないクライアントからのパスワード変更要求を受け付けることができます。詳細は、kadmind(1M) のマニュアルページを参照してください。
- 再実行キャッシュのデフォルトの場所が、RAM ベースのファイルシステムから /var/krb5/rcache/ の持続的記憶領域に移動しました。新しい場所では、システムがリブートされた場合に再実行から保護されます。rcache コードに対してパフォーマンスが強化されました。ただし、固定域の使用によって、再実行キャッシュ全体のパフォーマンスは落ちる場合があります。
- 再実行キャッシュをファイルまたはメモリのみの記憶域を使用するように構成することができます。鍵テーブルおよび資格キャッシュの種類または場所に対して構成可能な環境変数の詳細については、krb5envvar(5) のマニュアルページを参照してください。
- GSS 資格テーブルが Kerberos の GSS メカニズムで必要ではなくなりました。詳細は、424 ページの「GSS 資格の UNIX 資格へのマッピング」、または gsscred(1M)、gssd(1M)、および gsscred.conf(4) のマニュアルページを参照してください。
- Kerberos ユーティリティーの kinit と ktutil が、MIT Kerberos バージョン 1.2.1 に準拠するようになりました。この変更により、kinit コマンドに新しいオプションが追加され、ktutil コマンドに新しいサブコマンドが追加されました。詳細は、kinit(1) および ktutil(1) のマニュアルページを参照してください。
- Oracle Solaris の Kerberos 鍵配布センター (KDC) および kadmind が、MIT の Kerberos バージョン 1.2.1 ベースに基づいて変更されました。KDC では、現在のハッシュベースのデータベースよりも高い信頼性を備えた二分木ベースのデータベースがデフォルトで使用されます。詳細は、kdb5_util(1M) のマニュアルページを参照してください。

- `kpropd`、 `kadmind`、 `krb5kdc` および `ktkt_warnd` デーモンがサービス管理機能によって管理されます。このサービスに関する有効化、無効化、再起動などの管理アクションは `svcadm` コマンドを使用して実行できます。すべてのデーモンのサービスの状態は、`svcs` コマンドを使用して照会することができます。サービス管理機能の概要については、『Solaris のシステム管理 (基本編)』の第 18 章「サービスの管理 (概要)」を参照してください。

Solaris 9 の Kerberos 構成要素

Solaris 9 には、遠隔アプリケーションを除いて、410 ページの「Kerberos の構成要素」の構成要素がすべて含まれています。

SEAM 1.0.2 の構成要素

SEAM 1.0.2 には、遠隔アプリケーションが含まれています。SEAM 1.0 の構成要素のうちで、Solaris 9 リリースに組み込まれていないのはこれらのアプリケーションだけです。遠隔アプリケーションの構成要素は次のとおりです。

- クライアントアプリケーション – `ftp`、 `rcp`、 `rlogin`、 `rsh`、 および `telnet`
- サーバーデーモン – `ftpd`、 `rlogind`、 `rshd`、 および `telnetd`

Solaris 8 の Kerberos 構成要素

Solaris 8 に含まれている Kerberos サービスはクライアント側の部分だけで、Kerberos サービスの構成要素の多くは含まれていません。Solaris 8 が動作するシステムであれば、SEAM 1.0.1 を別にインストールしなくても Kerberos クライアントとしては動作します。これらのクライアント機能を使用するには、Solaris Easy Access Server 3.0 または Solaris 8 Admin Pack、 MIT ディストリビューション、あるいは Windows 2000 を使用する KDC をインストールする必要があります。チケットを配布するための構成済み KDC がないと、クライアント側の構成要素は十分に機能しません。このリリースには、次の構成要素が含まれています。

- チケットを取得、表示、破棄するユーザープログラム – `kinit`、 `klist`、 および `kdestroy`。
- Kerberos パスワードを変更するユーザープログラム – `kpasswd`。
- 鍵テーブル管理ユーティリティー – `ktutil`。
- PAM の拡張 – アプリケーションはさまざまな認証メカニズムを使用できます。PAM を使用すると、ログインとログアウトをユーザーが意識する必要をなくすことができます。
- GSS_API プラグイン – Kerberos プロトコルおよび暗号サポートを提供します。
- NFS クライアントおよびサーバーのサポート。

SEAM 1.0.1 の構成要素

SEAM 1.0.1 には、Solaris 8 に含まれていない SEAM 1.0 の構成要素がすべて含まれています。次の構成要素が含まれています。

- 鍵配布センター (KDC) (マスター):
  - Kerberos データベース管理デーモン – `kadmind`
  - Kerberos チケット処理デーモン – `krb5kdc`
- スレーブ KDC。
- データベース管理プログラム – `kadmin`、 `kadmin.local`。
- データベース伝播ソフトウェア – `kprop`。
- 遠隔アプリケーション – `ftp`、 `rcp`、 `rlogin`、 `rsh`、 および `telnet`。
- 遠隔アプリケーションデーモン – `ftpd`、 `rlogind`、 `rshd`、 および `telnetd`。
- 管理ユーティリティー – `kdb5_util`。
- Kerberos グラフィカル管理ツール (`gkadmin`) – 主体および主体ポリシーを管理することができます。この Java テクノロジベースの GUI は、`kadmin` コマンドに代わる機能です。
- 事前構成手順 – SEAM 1.0.1 のインストールおよび構成のパラメータを設定することによって、SEAM インストールを自動化できます。この手順は、特に複数のインストールを行うときに適しています。
- いくつかのライブラリ。

SEAM 1.0 の構成要素

SEAM 1.0 リリースには、410 ページの「Kerberos の構成要素」のすべての項目のほか、次の項目が含まれています。

- ユーティリティー (`gsscred`) とデーモン (`gssd`) – これらのプログラムは、UNIX のユーザー ID (UID) と主体名のマッピングに役立ちます。これらのプログラムが必要なのは、NFS サーバーがユーザーを識別するときに、主体名ではなく UNIX UID を使用しており、主体名と UNIX UID は形式が異なっているためです。
- Generic Security Service Application Programming Interface (GSS-API) – アプリケーションは、この API を利用して、複数のセキュリティーメカニズムを使用できます。新しいメカニズムを追加するたびに、アプリケーションをコンパイルし直す必要がありません。GSS-API はマシンに依存しないため、インターネット上のアプリケーションに適しています。GSS-API を使用すると、認証サービスだけでなく、整合性およびプライバシセキュリティーサービスをアプリケーションに組み込むことができます。

- RPCSEC_GSS Application Programming Interface (API) – NFS サービスが Kerberos 認証を使用することができます。RPCSEC_GSS は、使用しているメカニズムに依存しないセキュリティーサービスを提供するセキュリティー様式です。RPCSEC_GSS は、GSS-API 層の最上位に位置しています。GSS_API ベースのセキュリティーメカニズムは、プラグイン可能なので、RPCSEC_GSS を使用するアプリケーションで使用できます。
- 事前構成手順 – SEAM 1.0 のインストールおよび構成のパラメータを設定することによって、インストールを自動化できます。この手順は、特に複数のインストールを行うときに適しています。

◆ ◆ ◆ 第 22 章

22 Kerberos サービスの計画

この章は、Kerberos サービスのインストールと保守を行うシステム管理者を対象としています。この章では、Kerberos サービスをインストールまたは構成する前に、システム管理者が解決しておく必要があるインストールと構成の項目について説明します。

システム管理者やテクニカルサポート担当者が検討する必要がある項目は次のとおりです。

Kerberos の配備を計画する理由

Kerberos サービスをインストールする前に、いくつかの構成についての問題を解決する必要があります。初期インストール後に構成を変更することは不可能ではありませんが、変更によっては実装が困難な場合があります。また、変更によっては、KDCを再構築しなければならないことがあります。このため、Kerberos の構成を計画するときは、長期的な目標を考慮することをお勧めします。

Kerberos の基盤の配備には、KDC のインストール、ホストの鍵の作成、ユーザーの移行などの作業が含まれます。Kerberos の配備の再構成は最初の配備と同じくらいの労力を要するので、入念に計画し、再構成しなければならない事態を避けるようにしてください。

Kerberos レルムの計画

レルム は、ドメインに似た論理ネットワークです。レルムは、同一マスター KDC に登録されるシステムのグループを定義します。DNS ドメイン名を設定する場合と同様に、レルム名、レルムの数、および各レルムの大きさは、Kerberos サービスを構成する前に解決する必要があります。また、レルム間認証を行う場合は、レルム間の関係も定義する必要があります。

レルム名

レルム名には、任意の ASCII 文字列を使用できます。レルム名には通常、DNS ドメイン名と同じ名前を指定します。違いはレルム名は大文字で指定することです。この命名規則を利用すると、すでに使い慣れている名前を使用しながら、Kerberos サービスのレルム名と DNS 名前空間のドメイン名を区別することができます。DNS を使用しない場合、または別の文字列を使用する場合は、任意の文字列を使用できます。ただし、構成プロセスがより複雑になります。レルム名を付けるときは、標準のインターネット命名構造に準拠することをお勧めします。

レルムの数

インストールするレルムの数は、次の要因によって異なります。

- サポートするクライアント数。1 つのレルムに配置するクライアントが多すぎると、管理が複雑になり、レルムの分割が必要になることがあります。サポートできるクライアント数は、主に次の要因によって決まります。
  - 各クライアントが生成する Kerberos トラフィックの量
  - 物理ネットワークの帯域幅
  - ホストの処理速度

  インストールごとに制限が違ってくるため、最大クライアント数を決定する規則はありません。
- クライアント間の距離。クライアントが地理的に異なる領域に配置されている場合は、小さなレルムをいくつか設定することが望ましい方法です。
- KDC としてインストールできるホスト数。各レルムには、マスターサーバー用とスレーブサーバー用に、2 つ以上の KDC サーバーを持つべきです。

Kerberos レルムと管理ドメインがそろっているようにすることをお勧めします。Kerberos V レルムは、対応する DNS ドメインの複数のサブドメインにまたがることができます。

レルムの階層

複数のレルムを構成してレルム間認証を行う場合は、レルム間の接続方法を決定する必要があります。レルム間に階層関係を設定すると、関連付けたドメインに自動パスが作成されます。このとき、階層チェーン内のすべてのレルムが適切に構成されている必要があります。自動パスを利用すると、管理負荷を軽減することができます。ただし、ドメインのレベルが多い場合は、多くのトランザクションが発生するため、デフォルトのパスは使用しないことをお勧めします。

また、信頼関係を直接確立することもできます。直接の信頼関係は、2 つの階層レルム間にレベルが多すぎる場合または階層関係が設定されていない場合にもっとも有効です。直接接続は、使用するすべてのホストの `/etc/krb5/krb5.conf` ファイルに接続を定義する必要があります。このため、追加作業が必要になります。直接の信頼関係は、推移的関係とも呼ばれます。概要については、407 ページの「Kerberos レルム」を参照してください。複数のレルムを構成する手順については、449 ページの「レルム間認証の構成」を参照してください。

ホスト名のレルムへのマッピング

ホスト名のレルム名へのマッピングは、`krb5.conf` ファイルの `domain_realm` セクションに定義します。これらのマッピングは、必要に応じてドメイン全体およびホスト単位に定義できます。

DNS は、KDC に関する情報の検索にも使用できます。DNS を使用すると、変更を行うたびに全クライアントについて `krb5.conf` ファイルを編集する必要がないため、情報を変更しやすくなります。詳細は、`krb5.conf(4)` のマニュアルページを参照してください。

Solaris Express Developer Edition 1/08 および Solaris 10 5/08 リリースの時点で、Solaris Kerberos クライアントは Active Directory サーバーとより適切に相互運用できるようになりました。Active Directory サーバーは、ホストのマッピングにレルムを提供するように構成できます。

クライアントとサービス主体の名前

Kerberos サービスを使用しているときは、すべてのホストで DNS が有効である必要があります。DNS では、主体名に各ホストの完全指定のドメイン名 (FQDN) を含める必要があります。たとえば、ホスト名が `boston`、 DNS ドメイン名が `example.com`、 およびレルム名が `EXAMPLE.COM` の場合、ホストの主体名は `host/boston.example.com@EXAMPLE.COM` にするようにしてください。このドキュメントの例では、DNS を構成する必要があり、各ホストの FQDN を使用しています。

Kerberos サービスは DNS を介してホストの別名を正規化し、関連するサービスのサービス主体を構築する際には正規化された形式 (正規名) を使用します。そのため、サービス主体を作成する場合、サービス主体名のホスト名コンポーネントは、サービスをホストするシステムのホスト名の正規化形式にする必要があります。

次の例では、Kerberos サービスでホスト名を正規化する方法を示します。ユーザーがコマンド「ssh alpha.example.com」を実行するとします。ここで、`alpha.example.com` は正規名 `beta.example.com` の DNS ホストの別名です。ssh が Kerberos を呼び出し、`alpha.example.com` のホストサービスチケットを要求する場合、Kerberos サービスは、`alpha.example.com` を `beta.example.com` に正規化し、KDC からサービス主体「host/beta.example.com」のチケットを要求します。

ホストの FQDN を含む主体名は、`/etc/resolv.conf` ファイルの DNS ドメイン名を表す文字列と一致していることが重要です。Kerberos サービスでは、主体に FQDN を指定するときに、DNS ドメイン名は小文字にする必要があります。DNS ドメイン名には大文字と小文字を使用できますが、ホスト主体を作成する場合は小文字だけを使用します。たとえば、DNS ドメイン名には、`example.com` や `Example.COM` などの形式が使用できますが、ホストの主体名は、`host/boston.example.com@EXAMPLE.COM` でなければなりません。

また、サービス管理機能は、DNS クライアントサービスが実行されていない場合に多くのデーモンまたはコマンドが起動しないように構成されています。`kdb5_util`、`kadmind`、 `kpropd` デーモン、および `kprop` コマンドは、DNS サービスに依存するように構成されています。Kerberos サービスおよび SMF を使用して利用可能な機能を完全に活用するには、すべてのホスト上で DNS クライアントサービスを有効にする必要があります。

KDC と管理サービス用のポート

デフォルトでは、ポート `88` とポート `750` を KDC が使用し、ポート `749` を KDC 管理デーモンが使用します。別のポート番号を使用することもできます。ただし、ポート番号を変更する場合は、各クライアントの `/etc/services` および `/etc/krb5/krb5.conf` ファイルを変更する必要があります。また、これらのファイルに加えて、各 KDC の `/etc/krb5/kdc.conf` ファイルも更新する必要があります。

スレーブ KDC の数

スレーブ KDC は、マスター KDC と同様に、クライアントの資格を生成します。マスターが使用できなくなると、スレーブ KDC がバックアップとして使用されます。各レルムには、1 つ以上のスレーブ KDC が必要です。次の要因により、スレーブ KDC を追加する必要がある場合が考えられます。

- レルム内の物理セグメント数。通常は、レルム内のほかのセグメントが動作しない場合でも、少なくとも各セグメントで機能するように、ネットワークを設定する必要があります。この設定を実現するには、KDC をすべてのセグメントからアクセス可能にする必要があります。この場合、KDC はマスターまたはスレーブのどちらでも構いません。
- レルム内のクライアント数。スレーブ KDC サーバーを追加すると、現在のサーバーの負荷を軽減することができます。

スレーブ KDC の数に制限はありません。ただし、KDC データベースは、各サーバーに伝播する必要があります。このため、インストールした KDC サーバーが多くなるにつれて、レルム全体のデータ更新時間が長くなります。また、各スレーブには KDC データベースのコピーが保存されるため、スレーブが多くなるほど、セキュリティー侵害の危険性が高くなります。

1 つまたは複数のスレーブ KDC をマスター KDC と入れ替えするように構成することができます。このように 1 つ以上のスレーブ KDC をシステムに事前に構成しておくと、マスター KDC になんらかの理由で障害が発生した場合でも、マスター KDC と簡単に入れ替えすることができます。スワップ可能なスレーブ KDC の構成方法については、476 ページの「マスター KDC とスレーブ KDC の入れ替え」を参照してください。

GSS 資格の UNIX 資格へのマッピング

Kerberos サービスは、GSS 資格名から UNIX ユーザー ID (UID) へのマッピングを、NFS などこのマッピングを必要とする GSS アプリケーションのために提供します。GSS 資格名は Kerberos サービスを使用する場合の Kerberos 主体名と等価です。デフォルトのマッピングアルゴリズムでは、1 構成要素の Kerberos 主体名をとり、主体の一次名であるその構成要素を使用して、UID を検索します。検索は、デフォルトレルム、または `/etc/krb5/krb5.conf` の `auth_to_local_realm` パラメータを使用することで許可された任意のレルムで行われます。たとえば、ユーザー主体名 `bob@EXAMPLE.COM` は、パスワードテーブルを使用して、`bob` という名前の UNIX ユーザーの UID にマッピングされます。主体名が `admin` のインスタンスコンポーネントを含むため、ユーザー主体名 `bob/admin@EXAMPLE.COM` はマッピングされません。ユーザー資格のデフォルトマッピングが十分な場合、GSS 資格テーブルにデータを入れておく必要がありません。以前のリリースでは、NFS サービスを機能させるために、GSS 資格テーブルにデータを入れておく必要がありました。デフォルトマッピングでは十分でない場合、たとえばインスタンスコンポーネントを含む主体名をマッピングする場合などは、そのほかの方法を使用すべきです。詳細については、次のトピックを参照してください。

- 456 ページの「資格テーブルを作成する方法」
- 456 ページの「資格テーブルに 1 つのエントリを追加する方法」
- 457 ページの「レルム間の資格マッピングを提供する方法」
- 519 ページの「GSS 資格の UNIX 資格へのマッピングの監視」

Kerberos レルムへのユーザーの自動的な移行

PAM フレームワークを使用して、デフォルトの Kerberos レルムに有効なユーザーアカウントを持たない UNIX ユーザーを自動的に移行することができます。具体的には、`pam_krb5_migrate` モジュールを PAM サービスの認証スタックで使用します。Kerberos 主体を持たないユーザーがパスワードを使用してシステムへのログインを行うたびに、そのユーザーに対して Kerberos 主体が自動的に作成されるように、サービスを設定します。新しい主体パスワードは UNIX パスワードと同じにします。`pam_krb5_migrate` モジュールの使用方法については、472 ページの「Kerberos レルム内のユーザーを自動的に移行するように構成する方法」を参照してください。

使用するデータベースの伝播システム

マスター KDC に格納されているデータベースは、定期的にスレーブ KDC に伝播する必要があります。データベースの伝播は増分的に構成することができます。増分処理によって、データベース全体ではなく、更新された情報だけがスレーブ KDC に伝播されます。データベースの伝播の詳細については、481 ページの「Kerberos データベースの管理」を参照してください。

増分伝播を使用しない場合、最初に解決すべき問題の 1 つは、スレーブ KDC の更新頻度です。すべてのクライアントが最新の情報を使用できるようにしておく必要性について、更新の完了にかかる時間と比較検討する必要があります。

1 つのレルムに多くの KDC が配置されている場合は、1 つまたは複数のスレーブからもデータを伝播すると、伝播プロセスを並行して行うことができます。この方法を利用すると、データの更新時間は少なくなりますが、レルムの管理は複雑になります。この戦略の詳細については、494 ページの「並列伝播の設定」を参照してください。

レルム内でのクロックの同期

Kerberos 認証システムに参加するすべてのホストは、ホスト間の時刻の差が指定した最大時間内になるように内部クロックを同期化する必要があります。「クロックスキュー」と呼ばれるこの機能も、Kerberos セキュリティー検査の 1 つです。参加しているホスト間でクロックスキューを超過すると、要求が拒否されます。

すべてのクロックを同期化するときは、Network Time Protocol (NTP) ソフトウェアを使用します。詳細については、474 ページの「KDC と Kerberos クライアントのクロックの同期化」を参照してください。クロックの同期化にはほかにも方法があり、NTP は必須ではありません。任意の同期化形式を使用して、クロックスキューによるアクセス障害を回避してください。

クライアントの構成オプション

Solaris 10 の新機能の 1 つに、`kclient` 設定ユーティリティーがあります。このユーティリティーは、対話型モードまたは非対話型モードで実行できます。対話型モードでは、ユーザーは、Kerberos 固有のパラメータ値を要求されます。このパラメータ値によって、クライアントの設定時に既存のインストールを変更することができます。非対話型モードでは、前もってパラメータ値が設定されたファイルが使用されます。また、非対話型モードではコマンド行オプションも使用できます。対話型モードでも非対話型モードでも手動の処理よりも手順が少なくてすむので、プロセスが迅速化され、ミスが起こりにくくなります。

Solaris 10 5/08 リリースでは、構成不要の Kerberos クライアントが可能になるように変更されました。環境内で次の規則に従うと、Solaris Kerberos クライアントには明示的な構成手順が必要なくなります。

- DNS が、KDC 用の SRV レコードを返すように構成されている。
- レルム名が DNS ドメイン名に一致するか、または KDC がリフェラルをサポートする。
- Kerberos クライアントがキータブを必要としない。

場合によっては、Kerberos クライアントを明示的に構成する方がよいことがあります。

- リフェラルが使用されていない場合、構成不要のロジックは、レルムを決定するためにホストの DNS ドメイン名に依存します。これにより若干のセキュリティーリスクが導入されますが、このリスクは `dns_lookup_realm` を有効にするよりはるかに軽微です。
- `pam_krb5` モジュールは、キータブ内のホスト鍵のエントリに依存します。この要件は `krb5.conf` ファイルで無効にすることができますが、セキュリティー上の理由からそれはお勧めできません。`krb5.conf(4)` のマニュアルページを参照してください。
- 構成不要のプロセスは直接の構成に比べて非効率的であり、しかも DNS に大きく依存します。このプロセスは、直接構成されたクライアントに比べて多数の DNS 検索を実行します。

クライアントの設定プロセス全体の詳細については、460 ページの「Kerberos クライアントの構成」を参照してください。

クライアントログインのセキュリティーの改善

Solaris 10 11/06 リリースでは、クライアントのログイン時に `pam_krb5` モジュールを使用して、最新の TGT を発行した KDC と `/etc/krb5/krb5.keytab` に保存されているクライアントのホスト主体を発行した KDC が同じであることを確認します。`pam_krb5` モジュールは、認証スタックで構成されるときに KDC を確認します。クライアントのホスト主体を保存しない DHCP クライアントなど、一部の構成ではこの確認を無効にする必要があります。この確認をオフに設定するには、`krb5.conf` ファイルの `verify_ap_req_nofail` オプションを false に設定する必要があります。詳細は、470 ページの「チケット認可チケット (TGT) の確認を無効にする方法」を参照してください。

KDC の構成オプション

Solaris 10 5/08 リリースからは、LDAP を使用して Kerberos のデータベースファイルを管理する機能のサポートが追加されました。手順については、437 ページの「LDAP データサーバーを使用するように KDC を構成する方法」を参照してください。LDAP を使用すると、Solaris Kerberos データベースと既存の DS セットアップをより適切に調整する必要があるサイトの管理が簡略化されます。

Kerberos の暗号化タイプ

暗号化タイプとは、Kerberos サービスで使用される、暗号化アルゴリズム、暗号化モード、およびハッシュアルゴリズムを特定する識別子です。Kerberos サービスの鍵は、暗号化タイプに関連付けられ、サービスが鍵を使って暗号化処理を行うときに使用される暗号化アルゴリズムと暗号化モードを特定します。サポートされる暗号化タイプは次のとおりです。

- `des-cbc-md5`
- `des-cbc-crc`
- `des3-cbc-sha1-kd`
- `arcfour-hmac-md5`
- `arcfour-hmac-md5-exp`
- `aes128-cts-hmac-sha1-96`
- `aes256-cts-hmac-sha1-96`

注 – Solaris 10 8/07 より前のリリースでは、別売の Strong Cryptographic パッケージがインストールされている場合は、Kerberos サービスで `aes256-cts-hmac-sha1-96` 暗号化タイプを使用できます。

暗号化タイプを変更する場合は、新しい主体データベースの作成時に行います。KDC、サーバー、クライアント間の相互作用のために、既存のデータベースでの暗号化タイプの変更は困難です。データベースを再作成しない場合は、これらのパラメータは設定しないでおいてください。詳細については、590 ページの「Kerberos 暗号化タイプの使用」を参照してください。

注 – Solaris 10 が動作していないマスター KDC がインストールされている場合は、マスター KDC をアップグレードする前に、スレーブ KDC を Solaris 10 にアップグレードする必要があります。Solaris 10 のマスター KDC では、古いスレーブが処理できない新しい暗号化タイプが使用されます。

Kerberos グラフィカル管理ツールでのオンラインヘルプ URL

Kerberos グラフィカル管理ツール gkadmin ではオンラインヘルプ URL が使用されるため、「Help Contents」メニューが機能するように URL を適切に定義する必要があります。このマニュアルの HTML 版は、任意のサーバーにインストールできます。あるいは、http://www.oracle.com/technetwork/indexes/documentation/index.html のコレクションを使用することもできます。

URL は、Kerberos サービスを使用するようにホストを構成するときに krb5.conf ファイルで指定します。URL には、このドキュメントの「主体とポリシーの管理 (手順)」の「グラフィカルな Kerberos 管理ツール」を指定してください。必要に応じて、別の HTML ページを選択することもできます。

◆ ◆ ◆ 第 23 章

Kerberos サービスの構成 (手順)

この章では、KDC サーバー、ネットワークアプリケーションサーバー、NFS サーバー、および Kerberos クライアントの構成手順について説明します。手順の多くはスーパーユーザーアクセスを必要とするため、この作業はシステム管理者や上級ユーザーが行なってください。レルム間の構成手順など、KDC サーバーに関するトピックについても説明します。

この章の内容は次のとおりです。

Kerberos サービスの構成 (作業マップ)

構成手順は、その個々の手順がほかの手順に依存するため、特定の順序で実行する必要があります。多くの場合、これらの手順に従うことにより、Kerberos サービスに必要なサービスを設定できます。その他の手順は互いに依存しないため、任意のタイミングで実行できます。次の作業マップで、推奨する Kerberos のインストール順序を示します。

作業	説明	参照先
1. Kerberos インストールを計画します。	ソフトウェア構成処理を開始する前に、構成についての問題を解決します。事前に計画することで、結果的に時間やその他のリソースを節約できます。	第 22 章「Kerberos サービスの計画」
2. (省略可能) NTP をインストールします。	NTP ソフトウェアなどのクロック同期プロトコルを構成します。Kerberos サービスが正常に動作するには、レルムに含まれるすべてのシステムのクロックを同期化する必要があります。	474 ページの「KDC と Kerberos クライアントのクロックの同期化」
3. KDC サーバーを構成します。	レルムにマスター KDC サーバー、スレーブ KDC サーバー、および KDC データベースを構成および構築します。	431 ページの「KDC サーバーの構成」
4. (省略可能) KDC のセキュリティーを強化します。	KDC サーバーに対するセキュリティー侵害を回避します。	499 ページの「KDC サーバーへのアクセスを制限する方法」
5. (省略可能) 入れ替え可能な KDC を構成します。	マスター KDC とスレーブ KDC を簡単に入れ替えできるようにします。	476 ページの「入れ替え可能なスレーブ KDC を構成する方法」

追加の Kerberos サービスの構成 (作業マップ)

必要な手順が完了したあと、必要に応じて次の手順を行います。

作業	説明	参照先
レルム間認証を構成します。	レルム間の通信を使用可能にします。	449 ページの「レルム間認証の構成」
Kerberos アプリケーションサーバーを構成します。	サーバーが `ftp`、 `telnet`、 `rsh` などのサービスを Kerberos 認証を使ってサポートできるようにします。	452 ページの「Kerberos ネットワークアプリケーションサーバーの構成」
Kerberos クライアントを構成します。	クライアントが Kerberos サービスを使用できるようにします。	460 ページの「Kerberos クライアントの構成」
Kerberos NFS サーバーを構成します。	Kerberos 認証を必要とするファイルシステムを、サーバーが共有できるようにします。	454 ページの「Kerberos NFS サーバーの構成」
アプリケーションサーバーのセキュリティーを強化します。	認証されたトランザクションのみにアクセスを制限して、アプリケーションサーバーのセキュリティーを強化します。	499 ページの「Kerberos アプリケーションのみを有効にする方法」

KDC サーバーの構成

Kerberos ソフトウェアをインストールしたあと、KDC サーバーを構成する必要があります。資格を発行するには、1 つのマスター KDC と 1 つ以上のスレーブ KDC を構成する必要があります。KDC が発行する資格は、Kerberos サービスの基本要素であるため、KDC をインストールしないと、ほかの処理を行うことはできません。

マスター KDC とスレーブ KDC の最も大きな違いは、マスター KDC だけがデータベース管理要求を処理できることです。たとえば、パスワードの変更や新しい主体の追加は、マスター KDC で行います。これらの変更は、スレーブ KDC に伝播されます。資格の生成は、スレーブ KDC とマスター KDC が行います。この機能は、マスター KDC が応答できない場合に、冗長性を確保します。

表 23–1　KDC サーバーの構成 (作業マップ)

作業	説明	参照先
マスター KDC サーバーを構成します。	より複雑なインストールに必要な手動のプロセスを使用して、レルムにマスター KDC サーバーとデータベースを構成および構築します。	431 ページの「マスター KDC を手動で構成する方法」
	手動のプロセスを使用し、さらに KDC 用に LDAP を使用して、レルムにマスター KDC サーバーとデータベースを構成および構築します。	437 ページの「LDAP データサーバーを使用するように KDC を構成する方法」
スレーブ KDC サーバーを構成します。	より複雑なインストールに必要な手動のプロセスを使用して、レルムにスレーブ KDC サーバーを構成および構築します。	445 ページの「スレーブ KDC を手動で構成する方法」
KDC サーバー上の主体鍵を更新します。	新しい暗号化タイプを使用して KDC サーバー上のセッション鍵を更新します。	448 ページの「マスターサーバー上でチケット認可サービス鍵を更新する方法」

▼ マスター KDC を手動で構成する方法

この手順では、増分伝播を構成します。さらに、次の構成パラメータを使用します。

- レルム名 = `EXAMPLE.COM`
- DNS ドメイン名 = `example.com`
- マスター KDC = `kdc1.example.com`
- `admin` 主体 = `kws/admin`
- オンラインヘルプ URL = `http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956`

注 – この URL は「Kerberos グラフィカル管理ツール」のセクションを指すように調整してください (428 ページの「Kerberos グラフィカル管理ツールでのオンラインヘルプ URL」を参照)。

始める前に　この手順を実行するには、ホストが DNS を使用するように構成されている必要があります。マスター KDC を入れ替え可能にする場合の手順については、476 ページの「マスター KDC とスレーブ KDC の入れ替え」を参照してください。

1 マスター **KDC** 上でスーパーユーザーになります。

2 **Kerberos** 構成ファイル **(krb5.conf)** を編集します。

レルム名とサーバー名を変更する必要があります。このファイルの詳細は、`krb5.conf(4)` のマニュアルページを参照してください。

```
kdc1 # cat /etc/krb5/krb5.conf
[libdefaults]
        default_realm = EXAMPLE.COM

[realms]
        EXAMPLE.COM = {
        kdc = kdc1.example.com
        admin_server = kdc1.example.com
        }

[domain_realm]
        .example.com = EXAMPLE.COM
#
# if the domain name and realm name are equivalent,
# this entry is not needed
#
[logging]
        default = FILE:/var/krb5/kdc.log
        kdc = FILE:/var/krb5/kdc.log

[appdefaults]
    gkadmin = {
        help_url = http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956
        }
```

この例では、`default_realm`、 `kdc`、 `admin_server`、 およびすべての `domain_realm` エントリの行を変更しました。また、`help_url` を定義する行を編集しました。

注 – 暗号化タイプを制限する場合は、`default_tkt_enctypes` または `default_tgs_enctypes` の行を設定します。暗号化タイプの制限に関する詳細は、590 ページの「Kerberos 暗号化タイプの使用」を参照してください。

3 **KDC 構成ファイル (kdc.conf) を編集します。**

レルム名を変更する必要があります。このファイルの詳細は、kdc.conf(4) のマニュアルページを参照してください。

```
kdc1 # cat /etc/krb5/kdc.conf
[kdcdefaults]
        kdc_ports = 88,750

[realms]
        EXAMPLE.COM = {
                profile = /etc/krb5/krb5.conf
                database_name = /var/krb5/principal
                admin_keytab = /etc/krb5/kadm5.keytab
                acl_file = /etc/krb5/kadm5.acl
                kadmind_port = 749
                max_life = 8h 0m 0s
                max_renewable_life = 7d 0h 0m 0s
                sunw_dbprop_enable = true
                sunw_dbprop_master_ulogsize = 1000
                }
```

この例では、realms セクションのレルム名定義を変更しました。また、realms セクションで、増分伝播できるようにする行とログ内に保持する KDC マスターの更新数を選択する行を追加しました。

注 – 暗号化タイプを制限する場合は、permitted_enctypes、supported_enctypes、または master_key_type の行を設定します。暗号化タイプの制限に関する詳細は、590 ページの「Kerberos 暗号化タイプの使用」を参照してください。

---

4 **kdb5_util コマンドを使用して、KDC データベースを作成します。**

kdb5_util は、KDC データベースを作成するコマンドです。-s オプションを指定すると、kadmind と krb5kdc デーモンが起動する前に、KDC の認証に使用される stash ファイルが作成されます。

```
kdc1 # /usr/sbin/kdb5_util create -s
Initializing database '/var/krb5/principal' for realm 'EXAMPLE.COM'
master key name 'K/M@EXAMPLE.COM'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:      <Type the key>
Re-enter KDC database master key to verify:      <Type it again>
```

5 **Kerberos アクセス制御リストファイル (kadm5.acl) を編集します。**

作成された /etc/krb5/kadm5.acl ファイルには、KDC を管理できる主体名がすべて含まれている必要があります。

```
kws/admin@EXAMPLE.COM   *
```

エントリにより、EXAMPLE.COM レルム内の kws/admin 主体に対して、KDC 内の主体またはポリシーを変更する機能が与えられます。デフォルトのインストールでは、アスタリスク (*) が指定され、すべての admin 主体に変更権限が与えられます。このデフォルトの指定では、セキュリティーが低下する可能性があります。そのた

め、admin 主体すべてを列挙すると、セキュリティーが向上します。詳細は、kadm5.acl(4) のマニュアルページを参照してください。

6 **kadmin.local** コマンドを起動し、主体を追加します。

次の手順では、Kerberos サービスで使用される主体を作成します。

```
kdc1 # /usr/sbin/kadmin.local
kadmin.local:
```

a. データベースに管理主体を追加します。

必要な数の admin 主体を追加できます。KDC 構成処理を完了するには、1 つ以上の admin 主体を追加する必要があります。この例では、kws/admin 主体を追加します。「kws」の代わりに、適切な主体名で置き換えてください。

```
kadmin.local: addprinc kws/admin
Enter password for principal kws/admin@EXAMPLE.COM:     <Type the password>
Re-enter password for principal kws/admin@EXAMPLE.COM:      <Type it again>
Principal "kws/admin@EXAMPLE.COM" created.
kadmin.local:
```

b. **kiprop** 主体を作成します。

kiprop 主体はマスター KDC からの更新の承認に使用されます。

```
kadmin.local: addprinc -randkey kiprop/kdc1.example.com
Principal "kiprop/kdc1.example.com@EXAMPLE.COM" created.
kadmin.local:
```

c. **kadmind** サービスのキータブファイルを作成します。

このコマンドシーケンスは、kadmin/<FQDN> および changepw/<FQDN> の主体エントリを保持する特別なキータブファイルを作成します。これらの主体は、kadmind サービスと、変更されるパスワードに必要です。主体のインスタンスがホスト名のときは、/etc/resolv.conf ファイル内のドメイン名が大文字であるか小文字であるかに関係なく、FQDN は小文字で指定する必要があります。kadmin/changepw 主体は、Solaris リリースが稼働していないクライアントからのパスワードを変更するために使用されます。

```
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/kdc1.example.com
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab changepw/kdc1.example.com
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
```

```
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/changepw
Entry for principal kadmin/changepw with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local:
```

d. マスター **KDC** サーバーの **kiprop** 主体を **kadmind** キータブファイルに追加します。

kadm5.keytab ファイルに kiprop 主体を追加すると、増分伝播が開始されるときに、kadmind コマンドが自身を認証できます。

```
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kiprop/kdc1.example.com
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local:
```

e. **kadmin.local** を終了します。

以降の手順で必要になる主体をすべて追加しました。

```
kadmin.local: quit
```

7 **Kerberos** デーモンを起動します。

```
kdc1 # svcadm enable -r network/security/krb5kdc
kdc1 # svcadm enable -r network/security/kadmin
```

8 **kadmin** を起動して、主体をさらに追加します。

この時点で、Kerberos グラフィカル管理ツールを使用して、主体を追加できます。追加するには、上記の手順で作成した admin 主体名を使用してログインする必要があります。ただし、次のコマンド行の例では、簡略化されています。

```
kdc1 # /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin:
```

a. マスター **KDC** の **host** 主体を作成します。

ホスト主体は、変更をスレーブ KDC に伝播するために、kprop などの Kerberos アプリケーションで使用されます。この主体はまた、ssh などのアプリケーションを使用して、KDC サーバーにセキュリティー保護された遠隔アクセスを提供するためにも使用されます。主体のインスタンスがホスト名のときは、/etc/resolv.conf ファイル内のドメイン名が大文字であるか小文字であるかに関係なく、FQDN は小文字で指定する必要があります。

```
kadmin: addprinc -randkey host/kdc1.example.com
Principal "host/kdc1.example.com@EXAMPLE.COM" created.
kadmin:
```

b. **(省略可能) kclient** 主体を作成します。

この主体は、Kerberos クライアントのインストール中に kclient ユーティリティーで使用されます。このユーティリティーを使用しない場合は、主体に追加する必要はありません。kclient ユーティリティーのユーザーは、このパスワードを使用する必要があります。

```
kadmin: addprinc clntconfig/admin
Enter password for principal clntconfig/admin@EXAMPLE.COM:     <Type the password>
Re-enter password for principal clntconfig/admin@EXAMPLE.COM:      <Type it again>
Principal "clntconfig/admin@EXAMPLE.COM" created.
kadmin:
```

c. マスター **KDC** のキータブファイルにマスター **KDC** の **host** 主体を追加します。

キータブファイルにホスト主体を追加すると、sshd などのアプリケーションサーバーによってこの主体が自動的に使用されるようになります。

```
kadmin: ktadd host/kdc1.example.com
Entry for principal host/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

d. **kadmin** を終了します。

```
kadmin: quit
```

9 **(省略可能) NTP** などのクロック同期メカニズムを使用して、マスター **KDC** のクロックを同期化します。

Network Time Protocol (NTP) のインストールと使用は必要はありません。ただし、認証が正常終了するには、`krb5.conf` ファイルの `libdefaults` セクションに定義されているデフォルト時間内に収まるよう、すべてのクロックを調整する必要があります。NTP については、474 ページの「KDC と Kerberos クライアントのクロックの同期化」を参照してください。

10 スレーブ **KDC** の構成

冗長性を確保するには、スレーブ KDC を必ず 1 つ以上インストールするようにしてください。 手順については、445 ページの「スレーブ KDC を手動で構成する方法」を参照してください。

▼ LDAP データサーバーを使用するように KDC を構成する方法

Solaris 10 5/08 リリースから、次の手順を使用して、LDAP データサーバーを使用するように KDC を構成できるようになりました。

この手順では、次の構成パラメータを使用します。

- レルム名 = `EXAMPLE.COM`
- DNS ドメイン名 = `example.com`
- マスター KDC = `kdc1.example.com`
- ディレクトリサーバー = `dsserver.example.com`
- `admin` 主体 = `kws/admin`
- LDAP サービスの FMRI = `svc:/application/sun/ds:ds--var-opt-SUNWdsee-dsins1`
- オンラインヘルプ URL = `http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956`

注 – この URL は「Kerberos グラフィカル管理ツール」のセクションを指すように調整してください (428 ページの「Kerberos グラフィカル管理ツールでのオンラインヘルプ URL」を参照)。

始める前に この手順を実行する場合も、ホストが DNS を使用するように構成されている必要があります。パフォーマンスを向上させるためには、KDC と LDAP ディレクトリサービスを同じサーバーにインストールしてください。さらに、ディレクトリ

サーバーも稼働させるようにしてください。次の手順は、Sun Java Directory Server Enterprise Edition リリースを使用しているサーバーで機能します。

1 **KDC** 上でスーパーユーザーになります。

2 ディレクトリサーバーの証明書を作成し、その証明書をインポートします。

次の手順では、Directory Server 6.1 自己署名付き証明書を使用するように S10 KDC を構成します。証明書の期限が切れている場合は、『Sun Java System Directory Server Enterprise Edition 6.2 管理ガイド』の「自己署名済み証明書を管理する」の手順に従って証明書を更新してください。

a. 自己署名付きのディレクトリサーバー証明書をエクスポートします。

```
# /usr/sfw/bin/certutil -L -n defaultCert -d /export/sun-ds6.1/directory/alias \
        -P 'slapd-' -a > /var/tmp/ds_cert.pem
```

b. ローカルの証明書データベースを作成します。

```
# /usr/sfw/bin/certutil -N -d /var/ldap
```

c. ディレクトリサーバー証明書をローカルの証明書データベースに追加します。

```
# /usr/sfw/bin/certutil -A -n defaultCert -i /var/tmp/ds_cert -a -t CT -d /var/ldap
```

d. ディレクトリサーバー証明書をインポートします。

```
# pktool setpin keystore=nss dir=/var/ldap
# chmod a+r /var/ldap/*.db
# pktool import keystore=nss objtype=cert trust="CT" infile=/tmp/defaultCert.certutil.der \
        label=defaultCert dir=/var/ldap
```

3 必要に応じて、**LDAP** ディレクトリにデータを設定します。

4 既存のスキーマに **Kerberos** スキーマを追加します。

```
# ldapmodify -h dsserver.example.com -D "cn=directory manager" -f /usr/share/lib/ldif/kerberos.ldif
```

5 **LDAP** ディレクトリ内に **Kerberos** コンテナを作成します。

`krb5.conf` ファイルに次のエントリを追加します。

a. データベースの種類を定義します。

`realms` セクションに `database_module` を定義するエントリを追加します。

```
database_module = LDAP
```

b. データベースのモジュールを定義します。

```
[dbmodules]
    LDAP = {
        ldap_kerberos_container_dn = "cn=krbcontainer,dc=example,dc=com"
        db_library = kldap
        ldap_kdc_dn = "cn=kdc service,ou=profile,dc=example,dc=com"
```

```
        ldap_kadmind_dn = "cn=kadmin service,ou=profile,dc=example,dc=com"
        ldap_cert_path = /var/ldap
        ldap_servers = ldaps://dsserver.example.com
    }
```

c. **LDAP** ディレクトリ内に **KDC** を作成します。

このコマンドによって、krbcontainer とその他のいくつかのオブジェクトが作成されます。また、/var/krb5/.k5.EXAMPLE.COM マスター鍵の stash ファイルも作成されます。

```
# kdb5_ldap_util -D "cn=directory manager" create -P abcd1234 -r EXAMPLE.COM -s
```

6 **KDC** のバインド識別名 **(DN)** のパスワードを隠します。

これらのパスワードは、DS にバインドするときに KDC によって使用されます。KDC は、その KDC が使用しているアクセスの種類に応じて異なる役割を使用します。

```
# kdb5_ldap_util stashsrvpw "cn=kdc service,ou=profile,dc=example,dc=com"
# kdb5_ldap_util stashsrvpw "cn=kadmin service,ou=profile,dc=example,dc=com"
```

7 **KDC** サービスの役割を追加します。

a. 次に示すような内容を含む **kdc_roles.ldif** ファイルを作成します。

```
dn: cn=kdc service,ou=profile,dc=example,dc=com
cn: kdc service
sn: kdc service
objectclass: top
objectclass: person
userpassword: test123

dn: cn=kadmin service,ou=profile,dc=example,dc=com
cn: kadmin service
sn: kadmin service
objectclass: top
objectclass: person
userpassword: test123
```

b. **LDAP** ディレクトリ内に役割のエントリを作成します。

```
# ldapmodify -a -h dsserver.example.com -D "cn=directory manager" -f kdc_roles.ldif
```

8 **KDC** に関連する役割の **ACL** を設定します。

```
# cat << EOF | ldapmodify -h dsserver.example.com -D "cn=directory manager"
# Set kadmin ACL for everything under krbcontainer.
dn: cn=krbcontainer,dc=example,dc=com
changetype: modify
add: aci
aci: (target="ldap:///cn=krbcontainer,dc=example,dc=com")(targetattr="krb*")(version 3.0;\
      acl kadmin_ACL; allow (all)\
      userdn = "ldap:///cn=kadmin service,ou=profile,dc=example,dc=com";)

# Set kadmin ACL for everything under the people subtree if there are
# mix-in entries for krb princs:
```

```
dn: ou=people,dc=example,dc=com
changetype: modify
add: aci
aci: (target="ldap:///ou=people,dc=example,dc=com")(targetattr="krb*")(version 3.0;\
      acl kadmin_ACL; allow (all)\
      userdn = "ldap:///cn=kadmin service,ou=profile,dc=example,dc=com";)
EOF
```

9 **Kerberos 構成ファイル (krb5.conf) を編集します。**

レルム名とサーバー名を変更する必要があります。このファイルの詳細は、krb5.conf(4) のマニュアルページを参照してください。

```
kdc1 # cat /etc/krb5/krb5.conf
[libdefaults]
        default_realm = EXAMPLE.COM

[realms]
        EXAMPLE.COM = {
        kdc = kdc1.example.com
        admin_server = kdc1.example.com
        }

[domain_realm]
        .example.com = EXAMPLE.COM
#
# if the domain name and realm name are equivalent,
# this entry is not needed
#
[logging]
        default = FILE:/var/krb5/kdc.log
        kdc = FILE:/var/krb5/kdc.log

[appdefaults]
    gkadmin = {
        help_url = http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956
        }
```

この例では、default_realm、 kdc、 admin_server、 およびすべての domain_realm エントリの行を変更しました。また、help_url を定義する行を編集しました。

注 – 暗号化タイプを制限する場合は、default_tkt_enctypes または default_tgs_enctypes の行を設定します。暗号化タイプの制限に関する詳細は、590 ページの「Kerberos 暗号化タイプの使用」を参照してください。

---

10 **KDC 構成ファイル (kdc.conf) を編集します。**

レルム名を変更する必要があります。このファイルの詳細は、kdc.conf(4) のマニュアルページを参照してください。

```
kdc1 # cat /etc/krb5/kdc.conf
[kdcdefaults]
        kdc_ports = 88,750

[realms]
        EXAMPLE.COM = {
```

```
profile = /etc/krb5/krb5.conf
database_name = /var/krb5/principal
admin_keytab = /etc/krb5/kadm5.keytab
acl_file = /etc/krb5/kadm5.acl
kadmind_port = 749
max_life = 8h 0m 0s
max_renewable_life = 7d 0h 0m 0s
sunw_dbprop_enable = true
sunw_dbprop_master_ulogsize = 1000
}
```

この例では、realms セクションのレルム名定義を変更しました。また、realms セクションで、増分伝播できるようにする行とログ内に保持する KDC マスターの更新数を選択する行を追加しました。

注 – 暗号化タイプを制限する場合は、permitted_enctypes、 supported_enctypes、 または master_key_type の行を設定します。暗号化タイプの制限に関する詳細は、590 ページの「Kerberos 暗号化タイプの使用」を参照してください。

---

11 **Kerberos アクセス制御リストファイル (kadm5.acl) を編集します。**

作成された /etc/krb5/kadm5.acl ファイルには、KDC を管理できる主体名がすべて含まれている必要があります。

```
kws/admin@EXAMPLE.COM    *
```

エントリにより、EXAMPLE.COM レルム内の kws/admin 主体に対して、KDC 内の主体またはポリシーを変更する機能が与えられます。デフォルトのインストールでは、アスタリスク (*) が指定され、すべての admin 主体に変更権限が与えられます。このデフォルトの指定では、セキュリティーが低下する可能性があります。そのため、admin 主体すべてを列挙すると、セキュリティーが向上します。詳細は、kadm5.acl(4) のマニュアルページを参照してください。

12 **kadmin.local コマンドを起動し、主体を追加します。**

次の手順では、Kerberos サービスで使用される主体を作成します。

```
kdc1 # /usr/sbin/kadmin.local
kadmin.local:
```

a. データベースに管理主体を追加します。

必要な数の admin 主体を追加できます。KDC 構成処理を完了するには、1 つ以上の admin 主体を追加する必要があります。この例では、kws/admin 主体を追加します。「kws」の代わりに、適切な主体名で置き換えてください。

```
kadmin.local: addprinc kws/admin
Enter password for principal kws/admin@EXAMPLE.COM:     <Type the password>
Re-enter password for principal kws/admin@EXAMPLE.COM:      <Type it again>
Principal "kws/admin@EXAMPLE.COM" created.
kadmin.local:
```

b. kadmind サービスのキータブファイルを作成します。

このコマンドシーケンスは、kadmin および changepw の主体エントリを保持する特別なキータブファイルを作成します。これらの主体は、kadmind サービスに必要です。主体のインスタンスがホスト名のときは、/etc/resolv.conf ファイル内のドメイン名が大文字であるか小文字であるかに関係なく、FQDN は小文字で指定する必要があります。

```
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/kdc1.example.com
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab changepw/kdc1.example.com
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/changepw
Entry for principal kadmin/changepw with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local:
```

c. kadmin.local を終了します。

以降の手順で必要になる主体をすべて追加しました。

```
kadmin.local: quit
```

13 (省略可能) Kerberos サービスの LDAP 依存性を構成します。

LDAP および KDC サーバーが同一のホスト上で稼働しており、LDAP サービスが SMF FMRI を使って構成されている場合は、Kerberos デーモンの LDAP サービスに対する依存性を追加します。これにより、LDAP サービスが再起動すると KDC サービスも再起動するようになります。

a. **krb5kdc** サービスへの依存性を追加します。

```
# svccfg -s security/krb5kdc
svc:/network/security/krb5kdc> addpg dsins1 dependency
svc:/network/security/krb5kdc> setprop dsins1/entities = \
    fmri: "svc:/application/sun/ds:ds--var-opt-SUNWdsee-dsins1"
svc:/network/security/krb5kdc> setprop dsins1/grouping = astring: "require_all"
svc:/network/security/krb5kdc> setprop dsins1/restart_on = astring: "restart"
svc:/network/security/krb5kdc> setprop dsins1/type = astring: "service"
svc:/network/security/krb5kdc> exit
```

b. **kadmin** サービスへの依存性を追加します。

```
# svccfg -s security/kadmin
svc:/network/security/kadmin> addpg dsins1 dependency
svc:/network/security/kadmin> setprop dsins1/entities =\
    fmri: "svc:/application/sun/ds:ds--var-opt-SUNWdsee-dsins1"
svc:/network/security/kadmin> setprop dsins1/grouping = astring: "require_all"
svc:/network/security/kadmin> setprop dsins1/restart_on = astring: "restart"
svc:/network/security/kadmin> setprop dsins1/type = astring: "service"
svc:/network/security/kadmin> exit
```

14 Kerberos デーモンを起動します。

```
kdc1 # svcadm enable -r network/security/krb5kdc
kdc1 # svcadm enable -r network/security/kadmin
```

15 **kadmin** を起動して、主体をさらに追加します。

この時点で、Kerberos グラフィカル管理ツールを使用して、主体を追加できます。追加するには、上記の手順で作成した `admin` 主体名を使用してログインする必要があります。ただし、次のコマンド行の例では、簡略化されています。

```
kdc1 # /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin:
```

a. マスター **KDC** の **host** 主体を作成します。

ホスト主体は、`klist` や `kprop` などの Kerberos アプリケーションで使用されます。クライアントは、認証された NFS ファイルシステムをマウントするときに、この主体を使用します。主体のインスタンスがホスト名のときは、`/etc/resolv.conf` ファイル内のドメイン名が大文字であるか小文字であるかに関係なく、FQDN は小文字で指定する必要があります。

```
kadmin: addprinc -randkey host/kdc1.example.com
Principal "host/kdc1.example.com@EXAMPLE.COM" created.
kadmin:
```

b. (省略可能) **kclient** 主体を作成します。

この主体は、Kerberos クライアントのインストール中に kclient ユーティリティーで使用されます。このユーティリティーを使用しない場合は、主体に追加する必要はありません。kclient ユーティリティーのユーザーは、このパスワードを使用する必要があります。

```
kadmin: addprinc clntconfig/admin
Enter password for principal clntconfig/admin@EXAMPLE.COM:     <Type the password>
Re-enter password for principal clntconfig/admin@EXAMPLE.COM:     <Type it again>
Principal "clntconfig/admin@EXAMPLE.COM" created.
kadmin:
```

c. マスター **KDC** のキータブファイルにマスター **KDC** の **host** 主体を追加します。

キータブファイルに追加したホスト主体が、自動的に使用されます。

```
kadmin: ktadd host/kdc1.example.com
Entry for principal host/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

d. **kadmin** を終了します。

```
kadmin: quit
```

16 (省略可能) **NTP** などのクロック同期メカニズムを使用して、マスター **KDC** のクロックを同期化します。

Network Time Protocol (NTP) のインストールと使用は必要はありません。ただし、認証が正常終了するには、krb5.conf ファイルの libdefaults セクションに定義されているデフォルト時間内に収まるよう、すべてのクロックを調整する必要があります。NTP については、474 ページの「KDC と Kerberos クライアントのクロックの同期化」を参照してください。

17 スレーブ **KDC** の構成

冗長性を確保するには、スレーブ KDC を必ず 1 つ以上インストールするようにしてください。 手順については、445 ページの「スレーブ KDC を手動で構成する方法」を参照してください。

▼ スレーブ KDC を手動で構成する方法

この手順では、kdc2 という名前の新しいスレーブ KDC を構成します。また、増分伝播も構成します。この手順では、次の構成パラメータを使用します。

- レルム名 = EXAMPLE.COM
- DNS ドメイン名 = example.com
- マスター KDC = kdc1.example.com
- スレーブ KDC = kdc2.example.com
- admin 主体 = kws/admin

始める前に　マスター KDC が構成済みである必要があります。スレーブ KDC を入れ替え可能にする手順については、476 ページの「マスター KDC とスレーブ KDC の入れ替え」を参照してください。

1. マスター **KDC** 上でスーパーユーザーになります。

2. マスター **KDC** 上で **kadmin** を起動します。

   マスター KDC を構成するときに作成した admin 主体名を使用して、ログインする必要があります。

   ```
   kdc1 # /usr/sbin/kadmin -p kws/admin
   Enter password:      <Type kws/admin password>
   kadmin:
   ```

   a. マスター **KDC** のデータベースにスレーブホスト主体が存在しない場合は、追加します。

      スレーブが機能するには、ホスト主体が必要です。主体のインスタンスがホスト名のときは、/etc/resolv.conf ファイル内のドメイン名が大文字であるか小文字であるかに関係なく、FQDN は小文字で指定する必要があります。

      ```
      kadmin: addprinc -randkey host/kdc2.example.com
      Principal "host/kdc2.example.com@EXAMPLE.COM" created.
      kadmin:
      ```

   b. マスター **KDC** 上で、**kiprop** 主体を作成します。

      kiprop 主体は、マスター KDC からの増分伝播を承認するために使用されます。

      ```
      kadmin: addprinc -randkey kiprop/kdc2.example.com
      Principal "kiprop/kdc2.example.com@EXAMPLE.COM" created.
      kadmin:
      ```

   c. **kadmin** を終了します。

      ```
      kadmin: quit
      ```

3 マスター KDC 上で、Kerberos 構成ファイル (krb5.conf) を編集します。

各スレーブ用にエントリを追加する必要があります。このファイルの詳細は、krb5.conf(4) のマニュアルページを参照してください。

```
kdc1 # cat /etc/krb5/krb5.conf
 .
 .
[realms]
                EXAMPLE.COM = {
                kdc = kdc1.example.com
                kdc = kdc2.example.com
                admin_server = kdc1.example.com
        }
```

4 マスター KDC 上で、kadm5.acl に kiprop エントリを追加します。

このエントリにより、マスター KDC が kdc2 サーバー用の増分伝播の要求を受け取ることができるようになります。

```
kdc1 # cat /etc/krb5/kadm5.acl
*/admin@EXAMPLE.COM *
kiprop/kdc2.example.com@EXAMPLE.COM p
```

5 マスター KDC 上で、kadmind を再起動し、kadm5.acl ファイルの新しいエントリを使用します。

```
kdc1 # svcadm restart network/security/kadmin
```

6 すべてのスレーブ KDC 上で、KDC 管理ファイルをマスター KDC サーバーからコピーします。

この手順は、マスター KDC サーバーが、各 KDC サーバーに必要な情報を更新したため、すべてのスレーブ KDC 上で実行する必要があります。ftp などの転送メカニズムを使用して、マスター KDC から次のファイルのコピーを取得できます。

- /etc/krb5/krb5.conf
- /etc/krb5/kdc.conf

7 すべてのスレーブ KDC 上で、マスター KDC のエントリと各スレーブ KDC のエントリをデータベース伝播構成ファイル kpropd.acl に追加します。

この情報は、すべてのスレーブ KDC サーバー上で更新する必要があります。

```
kdc2 # cat /etc/krb5/kpropd.acl
host/kdc1.example.com@EXAMPLE.COM
host/kdc2.example.com@EXAMPLE.COM
```

8 すべてのスレーブ KDC 上で、Kerberos アクセス制御リストファイル kadm5.acl が反映されていないことを確認してください。

修正前の kadm5.acl ファイルは次のようになっています。

```
kdc2 # cat /etc/krb5/kadm5.acl
*/admin@___default_realm___ *
```

ファイルに kiprop のエントリがある場合は、それを削除します。

9 新しいスレーブ上で、**kdc.conf** のエントリを変更します。

sunw_dbprop_master_ulogsize エントリを sunw_dbprop_slave_poll を定義するエントリと置き換えます。エントリは、ポーリング時間を 2 分に設定します。

```
kdc1 # cat /etc/krb5/kdc.conf
[kdcdefaults]
        kdc_ports = 88,750

[realms]
        EXAMPLE.COM= {
                profile = /etc/krb5/krb5.conf
                database_name = /var/krb5/principal
                admin_keytab = /etc/krb5/kadm5.keytab
                acl_file = /etc/krb5/kadm5.acl
                kadmind_port = 749
                max_life = 8h 0m 0s
                max_renewable_life = 7d 0h 0m 0s
                sunw_dbprop_enable = true
                sunw_dbprop_slave_poll = 2m
        }
```

10 新しいスレーブ上で、**kadmin** コマンドを起動します。

マスター KDC を構成するときに作成した admin 主体名を使用して、ログインする必要があります。

```
kdc2 # /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin:
```

a. **kadmin** を使用して、スレーブのホスト主体をスレーブのキータブファイルに追加します。

このエントリにより kprop などの Kerberos アプリケーションが機能します。主体のインスタンスがホスト名のときは、/etc/resolv.conf ファイル内のドメイン名が大文字であるか小文字であるかに関係なく、FQDN は小文字で指定する必要があります。

```
kadmin: ktadd host/kdc2.example.com
Entry for principal host/kdc2.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc2.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc2.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc2.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc2.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

b. スレーブ **KDC** のキータブファイルに **kiprop** 主体を追加します。

krb5.keytab に kiprop 主体を追加すると、増分伝播が開始されるときに、kprpod コマンドが自身を認証できるようになります。

```
kadmin: ktadd kiprop/kdc2.example.com
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type AES-256 CTS mode
```

```
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

c. **kadmin** を終了します。

```
kadmin: quit
```

11 新しいスレーブ上で、**Kerberos** 伝播デーモンを起動します。

```
kdc2 # /usr/lib/krb5/kpropd
```

12 新しいスレーブ上で、**kdb5_util** を使用して **stash** ファイルを作成します。

```
kdc2 # /usr/sbin/kdb5_util stash
kdb5_util: Cannot find/read stored master key while reading master key
kdb5_util: Warning: proceeding without master key

Enter KDC database master key:      <Type the key>
```

13 **Kerberos** 伝播デーモンを終了します。

```
kdc2 # pkill kpropd
```

14 **(省略可能)** 新しいスレーブ **KDC** 上で、**NTP** などのクロック同期メカニズムを使用して、マスター **KDC** のクロックを同期化します。

Network Time Protocol (NTP) のインストールと使用は必要はありません。ただし、認証が正常終了するには、`krb5.conf` ファイルの `libdefaults` セクションに定義されているデフォルト時間内に収まるよう、すべてのクロックを調整する必要があります。NTP については、474 ページの「KDC と Kerberos クライアントのクロックの同期化」を参照してください。

15 新しいスレーブ上で、**KDC** デーモン **(krb5kdc)** を起動します。

`krb5kdc` サービスが有効なときは、システムがスレーブとして構成されていると `kpropd` も起動します。

```
kdc2 # svcadm enable network/security/krb5kdc
```

▼ マスターサーバー上でチケット認可サービス鍵を更新する方法

Solaris 10 リリース以前に作成された KDC サービスで、チケット認可サービス (TGS) 主体が DES 鍵のみを持つ場合、この鍵により、チケット認可チケット (TGT) セッション鍵の暗号化タイプは DES に限定されます。より強力なほかの暗号化タイ

プにも対応するリリースに KDC をアップデートすると、KDC で生成されるすべてのセッション鍵について暗号化を強化できるようになります。ただし、既存の TGS 主体の鍵を新しい暗号化タイプに対応するよう更新しなければ、TGT セッション鍵は DES に限定されたままになります。次の手順では、この鍵を更新して、ほかの暗号化タイプも使用できるようにします。

- **TGS サービス主体鍵を更新します。**

```
kdc1 % /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin: cpw -randkey krbtgt/EXAMPLE.COM@EXAMPLE.COM
```

例 **23–1** マスターサーバーから主体鍵を更新する

KDC マスターに root としてログオンした場合、次のコマンドを使用して TGS サービス主体を更新できます。

```
kdc1 # kadmin.local -q 'cpw -randkey krbtgt/EXAMPLE.COM@EXAMPLE.COM'
```

レルム間認証の構成

複数のレルムを接続して、レルム間でユーザーを認証することができます。レルム間の認証は、2 つのレルム間で共有される秘密鍵を作成することによって実行されます。レルム間の関係には、階層関係または直接接続があります (421 ページの「レルムの階層」を参照)。

▼ 階層関係のレルム間認証を設定する方法

この手順の例では、ENG.EAST.EXAMPLE.COM レルムと EAST.EXAMPLE.COM レルムを使用します。レルム間認証は、双方向に確立されます。この手順は、2 つのレルムのマスター KDC 上で完了する必要があります。

始める前に　マスター KDC の各レルムが構成済みである必要があります。認証プロセスを十分にテストするには、複数の Kerberos クライアントが構成されている必要があります。

1. **最初のマスター KDC 上でスーパーユーザーになります。**
2. **2 つのレルムに対して、TGT のサービス主体を作成します。**
   マスター KDC を構成したときに作成した admin 主体名を使用して、ログインする必要があります。

```
# /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin: addprinc krbtgt/ENG.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM
```

```
Enter password for principal krgtgt/ENG.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM:      <Type password>
kadmin: addprinc krbtgt/EAST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM
Enter password for principal krgtgt/EAST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM:      <Type password>
kadmin: quit
```

注 – 各サービス主体のパスワードは、2 つの KDC で同一である必要があります。そのため、サービス主体 krbtgt/ENG.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM のパスワードは、2 つのレルムで同じである必要があります。

3 **Kerberos 構成ファイル (krb5.conf) にエントリを追加して、すべてのレルムのドメイン名を定義します。**

```
# cat /etc/krb5/krb5.conf
[libdefaults]
 .
 .
[domain_realm]
        .eng.east.example.com = ENG.EAST.EXAMPLE.COM
        .east.example.com = EAST.EXAMPLE.COM
```

この例では、ENG.EAST.EXAMPLE.COM レルムと EAST.EXAMPLE.COM レルムのドメイン名を定義しています。Kerberos 構成ファイルは先頭から末尾方向に検索されるため、サブドメインは最初に定義してください。

4 **Kerberos 構成ファイルをこのレルムのすべてのクライアントにコピーします。**

レルム間認証が動作するには、すべてのシステム (スレーブ KDC などのサーバーを含む) に新しいバージョンの Kerberos 構成ファイル (/etc/krb5/krb5.conf) がインストールされている必要があります。

5 もう一方のレルムで上記の全手順を繰り返します。

▼ 直接接続のレルム間認証を確立する方法

この手順では、ENG.EAST.EXAMPLE.COM レルムと SALES.WEST.EXAMPLE.COM レルムを使用します。レルム間認証は、双方向に確立されます。この手順は、2 つのレルムのマスター KDC 上で完了する必要があります。

始める前に マスター KDC の各レルムが構成済みである必要があります。認証プロセスを十分にテストするには、複数の Kerberos クライアントが構成されている必要があります。

1 **いずれかのマスター KDC サーバー上でスーパーユーザーになります。**

2 2つのレルムに対して、TGTのサービス主体を作成します。

マスターKDCを構成したときに作成したadmin主体名を使用して、ログインする必要があります。

```
# /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin: addprinc krbtgt/ENG.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM
Enter password for principal
  krgtgt/ENG.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM:      <Type the password>
kadmin: addprinc krbtgt/SALES.WEST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM
Enter password for principal
  krgtgt/SALES.WEST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM:      <Type the password>
kadmin: quit
```

注 – 各サービス主体のパスワードは、2つのKDCで同一である必要があります。そのため、サービス主体krbtgt/ENG.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COMのパスワードは、2つのレルムで同じである必要があります。

3 Kerberos構成ファイルにエントリを追加して、遠隔レルムへの直接パスを定義します。

この例は、ENG.EAST.EXAMPLE.COMレルムのクライアントを示しています。SALES.WEST.EXAMPLE.COMレルムで適切な定義をするには、レルム名を入れ替える必要があります。

```
# cat /etc/krb5/krb5.conf
[libdefaults]
 .
 .
[capaths]
    ENG.EAST.EXAMPLE.COM = {
        SALES.WEST.EXAMPLE.COM = .
    }

    SALES.WEST.EXAMPLE.COM = {
         ENG.EAST.EXAMPLE.COM = .
    }
```

4 Kerberos構成ファイルを現在のレルムのすべてのクライアントにコピーします。

レルム間認証が動作するには、すべてのシステム(スレーブKDCなどのサーバーを含む)に新しいバージョンのKerberos構成ファイル(/etc/krb5/krb5.conf)がインストールされている必要があります。

5 もう一方のレルムで上記の全手順を繰り返します。

Kerberos ネットワークアプリケーションサーバーの構成

ネットワークアプリケーションサーバーとは、次のいずれかのネットワークアプリケーションを 1 つ以上使ってアクセスを提供するホストのことです。 `ftp`、 `rcp`、 `rlogin`、 `rsh`、 `ssh`、 `telnet`。いくつかの手順を実行するだけで、これらのコマンドの Kerberos バージョンをサーバー上で有効にすることができます。

▼ Kerberos ネットワークアプリケーションサーバーを構成する方法

この手順では、次の構成パラメータを使用します。

- アプリケーションサーバー = `boston`
- `admin` 主体 = `kws/admin`
- DNS ドメイン名 = `example.com`
- レルム名 = `EXAMPLE.COM`

始める前に　この手順を行う場合には、マスター KDC がすでに構成されていなければなりません。このプロセスを十分にテストするには、複数の Kerberos クライアントが構成されている必要があります。

1. **(省略可能) NTP クライアントなどのクロック同期メカニズムをインストールします。**

   NTP については、474 ページの「KDC と Kerberos クライアントのクロックの同期化」を参照してください。

2. **新しいサーバーの主体を追加し、そのサーバーのキータブを更新します。**

   次のコマンドは、ホスト主体の存在有無を報告します。

   ```
   boston # klist -k |grep host
   4 host/boston.example.com@EXAMPLE.COM
   4 host/boston.example.com@EXAMPLE.COM
   4 host/boston.example.com@EXAMPLE.COM
   4 host/boston.example.com@EXAMPLE.COM
   ```

   このコマンドを実行しても主体が返されなかった場合、次の手順に従って新しい主体を作成します。

   Kerberos グラフィカル管理ツールを使って主体を追加する方法については、531 ページの「新しい Kerberos 主体を作成する方法」を参照してください。次の手順例

は、コマンド行から必要な主体を追加する方法を示したものです。マスター KDC を構成するときに作成した admin 主体名を使用して、ログインする必要があります。

```
boston # /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin:
```

a. サーバーの **host** 主体を作成します。

host 主体を使用する目的は次のとおりです。

- rsh や ssh など、遠隔コマンドの使用時にトラフィックを認証します。
- pam_krb5 により、host 主体を使用してユーザーの Kerberos 資格が信頼できる KDC から取得されたことを確認し、KDC へのなりすまし攻撃を防ぎます。
- root ユーザーが root 主体の存在なしで Kerberos 資格を自動的に取得できるようにします。これは、共有で Kerberos 資格が必要な場合に手動の NFS マウントを実行するのに役立つことがあります。

リモートアプリケーションを使用するトラフィックに Kerberos サービスによる認証が必要な場合は、この主体が必要です。サーバーに複数のホスト名が割り当てられている場合は、ホスト名の FQDN 形式を使用してホスト名ごとに主体を作成します。

```
kadmin: addprinc -randkey host/boston.example.com
Principal "host/boston.example.com" created.
kadmin:
```

b. サーバーの **host** 主体をサーバーのキータブに追加します。

kadmin コマンドが実行中でないときは、次のようにコマンドを再起動します。

/usr/sbin/kadmin -p *kws*/admin

サーバーに複数のホスト名が割り当てられている場合は、ホスト名ごとに主体をキータブに追加します。

```
kadmin: ktadd host/boston.example.com
Entry for principal host/boston.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/boston.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/boston.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/boston.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/boston.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

c. **kadmin** を終了します。

```
kadmin: quit
```

Kerberos NFS サーバーの構成

NFS サービスは UNIX ユーザー ID (UID) を使用してユーザーを識別しており、GSS 資格の主体を直接に使用することはできません。そのため、資格の主体を UID に対応付けるために、ユーザー資格の主体を UNIX UID に割り当てる資格テーブルを作成する必要があります。デフォルトの資格マッピングの詳細については、424 ページの「GSS 資格の UNIX 資格へのマッピング」を参照してください。この節では、Kerberos NFS サーバーの構成手順、資格テーブルの管理手順、および NFS マウントしたファイルシステムに対して Kerberos セキュリティーモードを有効にする手順を中心に説明します。次の表は、この節で説明する作業の一覧です。

表 23–2 Kerberos NFS サーバーの構成 (作業マップ)

作業	説明	参照先
Kerberos NFS サーバーを構成します。	Kerberos 認証を必要とするファイルシステムを、サーバーが共有できるようにします。	454 ページの「Kerberos NFS サーバーを構成する方法」
資格テーブルを作成します。	デフォルトのマッピングが十分でない場合、GSS 資格から UNIX ユーザー ID へのマッピングに使用できる資格テーブルを生成します。	456 ページの「資格テーブルを作成する方法」
ユーザー資格を UNIX UID に割り当てる資格テーブルを変更します。	資格テーブルの情報を更新します。	456 ページの「資格テーブルに 1 つのエントリを追加する方法」
類似する 2 つのレルム間の資格マッピングを作成します。	レルムがパスワードファイルを共有している場合に、あるレルムから別のレルムに UID をマッピングする方法を説明します。	457 ページの「レルム間の資格マッピングを提供する方法」
Kerberos 認証を使用してファイルシステムを共有します。	セキュリティーモードを使用してファイルシステムを共有し、Kerberos 認証を常に行います。	458 ページの「複数の Kerberos セキュリティーモードで安全な NFS 環境を設定する方法」

▼ Kerberos NFS サーバーを構成する方法

この手順では、次の構成パラメータを使用します。

- レルム名 = `EXAMPLE.COM`
- DNS ドメイン名 = `example.com`
- NFS サーバー = `denver.example.com`
- `admin` 主体 = `kws/admin`

1 **Kerberos NFS サーバーを構成するための前提条件を完了します。**

マスター KDC が構成済みである必要があります。処理を十分にテストするには、複数のクライアントが必要です。

2 (省略可能) NTP クライアントなどのクロック同期メカニズムをインストールします。

Network Time Protocol (NTP) のインストールと使用は必要はありません。ただし、認証が正常終了するには、すべてのクロックが、krb5.conf ファイル内の clockskew 関係指定子で定義されている最大の誤差以内で KDC サーバー上の時刻と同期化されている必要があります。NTP については、474 ページの「KDC と Kerberos クライアントのクロックの同期化」を参照してください。

3 NFS サーバーを Kerberos クライアントとして構成します。

460 ページの「Kerberos クライアントの構成」の手順に従ってください。

4 kadmin を起動します。

Kerberos グラフィカル管理ツールを使って主体を追加する方法については、531 ページの「新しい Kerberos 主体を作成する方法」を参照してください。追加するときは、マスター KDC を構成するときに作成した admin 主体名を使用してログインする必要があります。ただし、次の例では、コマンド行を使用して、必要な主体を追加しています。

```
denver # /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin:
```

a. サーバーの NFS サービス主体を作成します。

主体のインスタンスがホスト名のときは、/etc/resolv.conf ファイル内のドメイン名が大文字であるか小文字であるかに関係なく、FQDN は小文字で指定する必要があります。

NFS データへのアクセスに使用されるシステム上の一意のインタフェースごとに、上記の手順を繰り返します。ホストに一意の名前を持ったインタフェースが複数存在する場合、一意の名前は、それぞれに NFS サービス主体を持つ必要があります。

```
kadmin: addprinc -randkey nfs/denver.example.com
Principal "nfs/denver.example.com" created.
kadmin:
```

b. サーバーの NFS サービス主体をサーバーのキータブファイルに追加します。

手順 a で作成した一意のサービス主体ごとに、この手順を繰り返します。

```
kadmin: ktadd nfs/denver.example.com
Entry for principal nfs/denver.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs denver.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

c. **kadmin** を終了します。

```
kadmin: quit
```

5 **(省略可能) 必要に応じて、特殊な GSS 資格マップを作成します。**

通常、Kerberos サービスは、GSS 資格と UNIX UID 間の適切な対応表を生成します。デフォルトのマッピングについては、424 ページの「GSS 資格の UNIX 資格へのマッピング」を参照してください。デフォルトのマッピングが十分でない場合は、456 ページの「資格テーブルを作成する方法」を参照してください。

6 **NFS ファイルシステムを Kerberos セキュリティーモードで共有します。**

詳細は、458 ページの「複数の Kerberos セキュリティーモードで安全な NFS 環境を設定する方法」を参照してください。

▼ 資格テーブルを作成する方法

gsscred 資格テーブルは、Kerberos 主体を UID に割り当てるために NFS サーバーが使用します。デフォルトでは、主体名の一次の部分が UNIX のログイン名に一致します。NFS クライアントが Kerberos 認証を使用して NFS サーバーからファイルシステムをマウントするには、デフォルトのマッピングが十分でない場合、このテーブルを作成する必要があります。

1 **/etc/gss/gsscred.conf を編集してセキュリティーメカニズムを変更します。**

このメカニズムを files に変更します。

2 **gsscred コマンドを使用して資格テーブルを作成します。**

```
# gsscred -m kerberos_v5 -a
```

gsscred コマンドは、/etc/nsswitch.conf ファイル内の passwd エントリに指定されているすべてのソースから、情報を収集します。資格テーブルにローカルのパスワードエントリを入れたくない場合は、files エントリを一時的に削除しなければならないことがあります。詳細は、gsscred(1M) のマニュアルページを参照してください。

▼ 資格テーブルに 1 つのエントリを追加する方法

始める前に　この手順を行うには、gsscred テーブルがすでに NFS サーバーに作成済みである必要があります。手順については、456 ページの「資格テーブルを作成する方法」を参照してください。

1 **NFS サーバー上でスーパーユーザーになります。**

2 **gsscred** コマンドを使用して、エントリを資格テーブルに追加します。

 # gsscred -m *mech* **[-n** *name* **[-u** *uid* **]] -a**

mech 使用するセキュリティーメカニズムを定義します。

name KDC に定義されている、ユーザーの主体名を定義します。

uid パスワードデータベースに定義されている、ユーザーの UID を定義します。

-a UID を主体名の割り当てに追加します。

例 23–2 資格テーブルに複数の構成要素主体を追加する

次の例では、 sandy/admin という名前の主体にエントリを 1 つ追加し、UID 3736 に割り当てます。

```
# gsscred -m kerberos_v5 -n sandy/admin -u 3736 -a
```

例 23–3 異なるドメインの主体を資格テーブルに追加する

次の例では、sandy/admin@EXAMPLE.COM という名前の主体にエントリを 1 つ追加し、UID 3736 に割り当てます。

```
# gsscred -m kerberos_v5 -n sandy/admin@EXAMPLE.COM -u 3736 -a
```

▼ レルム間の資格マッピングを提供する方法

この手順では、同じパスワードファイルを使用するレルム間の適切な資格マッピングを提供します。この例では、CORP.EXAMPLE.COM レルムと SALES.EXAMPLE.COM レルムは同じパスワードファイル使用します。bob@CORP.EXAMPLE.COM と bob@SALES.EXAMPLE.COM の資格は同じ UID にマッピングされます。

1 スーパーユーザーになります。

2 クライアントシステム上で、エントリを **krb5.conf** ファイルに追加します。

```
# cat /etc/krb5/krb5.conf
[libdefaults]
        default_realm = CORP.EXAMPLE.COM
 .
[realms]
    CORP.EXAMPLE.COM = {
        .
        auth_to_local_realm = SALES.EXAMPLE.COM
        .
    }
```

例 23–4 同じパスワードファイルを使用してレルム間で資格をマッピングする

この例では、同じパスワードファイルを使用するレルム間の適切な資格マッピングを提供します。この例では、CORP.EXAMPLE.COM レルムと SALES.EXAMPLE.COM レルムは同じパスワードファイル使用します。bob@CORP.EXAMPLE.COM と bob@SALES.EXAMPLE.COM の資格は同じ UID にマッピングされます。クライアントシステム上で、エントリを krb5.conf ファイルに追加します。

```
# cat /etc/krb5/krb5.conf
[libdefaults]
        default_realm = CORP.EXAMPLE.COM
 .
[realms]
    CORP.EXAMPLE.COM = {
        .
        auth_to_local_realm = SALES.EXAMPLE.COM
        .
    }
```

注意事項 資格マッピングの問題を障害追跡する手順については、519 ページの「GSS 資格の UNIX 資格へのマッピングの監視」を参照してください。

▼ 複数の Kerberos セキュリティーモードで安全な NFS 環境を設定する方法

この手順により、さまざまなセキュリティーモードまたはフレーバを使用して、NFS サーバーが NFS に安全にアクセスできるようになります。クライアントがセキュリティーフレーバについて NFS サーバーとネゴシエーションを行うとき、クライアントがアクセスしたサーバーが提供する最初のフレーバが使用されます。このフレーバは、NFS サーバーが共有するファイルシステムにおける後続のクライアント要求すべてに使用されます。

1. **NFS サーバー上でスーパーユーザーになります。**

2. **キータブファイルに NFS サービス主体が存在することを検証します。**

   klist コマンドを指定すると、キータブファイルが存在するかどうかが出力され、その主体が表示されます。キータブファイルが存在しない場合、または NFS サービス主体が存在しない場合は、454 ページの「Kerberos NFS サーバーを構成する方法」のすべての手順が完了していることを検証する必要があります。

   ```
   # klist -k
   Keytab name: FILE:/etc/krb5/krb5.keytab
   KVNO Principal
   ---- --------------------------------------------------------
      3 nfs/denver.example.com@EXAMPLE.COM
      3 nfs/denver.example.com@EXAMPLE.COM
   ```

```
   3 nfs/denver.example.com@EXAMPLE.COM
   3 nfs/denver.example.com@EXAMPLE.COM
```

3 **/etc/nfssec.conf** ファイル内の **Kerberos** セキュリティーモードを有効にします。

/etc/nfssec.conf ファイルを編集して、Kerberos セキュリティーモードの先頭にある「#」を削除します。

```
# cat /etc/nfssec.conf
 .
 .
#
# Uncomment the following lines to use Kerberos V5 with NFS
#
krb5            390003  kerberos_v5     default -               # RPCSEC_GSS
krb5i           390004  kerberos_v5     default integrity       # RPCSEC_GSS
krb5p           390005  kerberos_v5     default privacy         # RPCSEC_GSS
```

4 **/etc/dfs/dfstab** ファイルを編集し、必要なセキュリティーモードを **sec=** オプションに指定して、適切なエントリに追加します。

share -F nfs -o sec=*mode file-system*

mode ファイルシステムを共有するときに使用するセキュリティーモードを指定します。複数のセキュリティーモードを使用するときは、デフォルトとして、リストの最初のモードが使用されます。

file-system 共有するファイルシステムへのパスを定義します。

指定されたファイルシステムのファイルにアクセスするすべてのクライアントは、Kerberos 認証が必要です。ファイルにアクセスするには、NFS クライアント上にユーザー主体が認証される必要があります。

5 **NFS** サービスがサーバー上で動作していることを確認します。

1 つまたは複数の share コマンドをはじめて実行する場合、NFS デーモンが動作していないことがあります。次のコマンドでデーモンを再起動します。

```
# svcadm restart network/nfs/server
```

6 (省略可能) オートマウンタを使用する場合は、**auto_master** データベースを編集して、デフォルト以外のセキュリティーモードを選択してください。

ファイルシステムのアクセスにオートマウンタを使用しない場合やデフォルトの選択をセキュリティーモードとして使用する場合は、この手順を行う必要はありません。

file-system auto_home -nosuid,sec=*mode*

7 (省略可能) 手動で **mount** コマンドを実行し、デフォルト以外のモードを使用してファイルシステムにアクセスします。

代わりに、mount コマンドにセキュリティーモードを指定できますが、オートマウンタは利用できません。

```
# mount -F nfs -o sec=mode file-system
```

例 **23–5** 1 つの Kerberos セキュリティーモードでファイルシステムを共有する

この例の dfstab ファイルの行は、NFS サービスを使用してファイルにアクセスするには、Kerberos 認証が成功する必要があることを示しています。

```
# grep krb /etc/dfs/dfstab
share -F nfs -o sec=krb5 /export/home
```

例 **23–6** 複数の Kerberos セキュリティーモードでファイルシステムを共有する

次の例では、3 つの Kerberos セキュリティーモードがすべて選択されています。クライアントと NFS サーバーの間で、どのモードを使用するかのネゴシエーションが行われます。コマンドの最初のモードが失敗すると、次のモードが試行されます。詳細は、nfssec(5) のマニュアルページを参照してください。

```
# grep krb /etc/dfs/dfstab
share -F nfs -o sec=krb5:krb5i:krb5p /export/home
```

Kerberos クライアントの構成

Kerberos クライアントは、Kerberos サービスを使用する同じネットワーク上のすべてのホスト (KDC サーバーを除く) です。この節では、Kerberos クライアントのインストール手順と、root 認証を使用して NFS ファイルシステムをマウントする方法について説明します。

Kerberos クライアントの構成 (作業マップ)

次の作業マップは、Kerberos クライアントの設定に関連するすべての手順が含まれます。各行には、作業識別名、その作業を行う理由、および作業へのリンクが含まれます。

作業	説明	参照先
Kerberos クライアントのインストールプロファイルを確立します。	Kerberos クライアントの自動インストールに使用されるクライアントのインストールプロファイルを生成します。	461 ページの「Kerberos クライアントのインストールプロファイルの作成方法」
Kerberos クライアントを構成します。	手動で Kerberos クライアントをインストールします。各クライアントのインストールに独自のインストールパラメータが必要な場合に、この手順を使用します。	464 ページの「Kerberos クライアントを手動で構成する方法」
	自動的に Kerberos クライアントをインストールします。各クライアントのインストールパラメータが同じ場合に、この手順を使用します。	462 ページの「Kerberos クライアントを自動的に構成する方法」
	対話的に Kerberos クライアントをインストールします。2、3 のインストールパラメータを変更する必要がある場合にだけ、この手順を使用します。	463 ページの「Kerberos クライアントを対話的に構成する方法」
クライアントが root ユーザーとして NFS ファイルシステムにアクセスすることを許可します	root アクセス付きで共有される NFS ファイルシステムをクライアントがマウントできるように、root 主体をクライアント上に作成します。また、cron ジョブを実行できるように、NFS ファイルシステムへの非対話的な root アクセスをクライアントがセットアップすることを許可します。	471 ページの「Kerberos によって保護された NFS ファイルシステムに root ユーザーとしてアクセスする方法」
クライアントのチケット認可チケット (TGT) を発行した KDC の確認を無効にします。	ローカルのキータブファイルにホスト主体を保存しないクライアントが、TGT を発行した KDC とホスト主体を発行した KDC が同じサーバーであることを確認するセキュリティー検査を省略できるようにします。	470 ページの「チケット認可チケット (TGT) の確認を無効にする方法」

▼ Kerberos クライアントのインストールプロファイルの作成方法

この手順は、Kerberos クライアントをインストールする際に使用される kclient プロファイルを作成します。kclient プロファイルを使用することにより、入力エラーの可能性を減らします。また、プロファイルを使用すると、対話型のプロセスと比べて、ユーザーの介入も減ります。

1 スーパーユーザーになります。

2 **kclient** インストールプロファイルを作成します。

kclient プロファイルの例は、次のようになります。

```
client# cat /net/denver.example.com/export/install/profile
REALM EXAMPLE.COM
KDC kdc1.example.com
ADMIN clntconfig
FILEPATH /net/denver.example.com/export/install/krb5.conf
NFS 1
DNSLOOKUP none
```

▼ Kerberos クライアントを自動的に構成する方法

始める前に　この手順では、インストールプロファイルを使用します。461 ページの「Kerberos クライアントのインストールプロファイルの作成方法」を参照してください。

1 スーパーユーザーになります。

2 **kclient** インストールスクリプトを実行します。

プロセスを完了するには、clntconfig 主体のパスワードを入力する必要があります。

```
client# /usr/sbin/kclient -p /net/denver.example.com/export/install/profile

Starting client setup
------------------------------------------------

kdc1.example.com

Setting up /etc/krb5/krb5.conf.

Obtaining TGT for clntconfig/admin ...
Password for clntconfig/admin@EXAMPLE.COM:      <Type the password>

nfs/client.example.com entry ADDED to KDC database.
nfs/client.example.com entry ADDED to keytab.

host/client.example.com entry ADDED to KDC database.
host/client.example.com entry ADDED to keytab.

Copied /net/denver.example.com/export/install/krb5.conf.

------------------------------------------------
Setup COMPLETE.

client#
```

例 23–7 コマンド行指定の優先を利用して Kerberos クライアントを自動的に構成する

次の例は、インストールプロファイルに設定されている DNSARG パラメータと KDC パラメータに対しコマンド行での指定が優先します。

```
# /usr/sbin/kclient -p /net/denver.example.com/export/install/profile\
-d dns_fallback -k kdc2.example.com

Starting client setup
-------------------------------------------------

kdc1.example.com

Setting up /etc/krb5/krb5.conf.

Obtaining TGT for clntconfig/admin ...
Password for clntconfig/admin@EXAMPLE.COM:      <Type the password>

nfs/client.example.com entry ADDED to KDC database.
nfs/client.example.com entry ADDED to keytab.

host/client.example.com entry ADDED to KDC database.
host/client.example.com entry ADDED to keytab.

Copied /net/denver.example.com/export/install/krb5.conf.

-------------------------------------------------
Setup COMPLETE.

client#
```

▼ Kerberos クライアントを対話的に構成する方法

この手順では、インストールプロファイルなしで kclient インストールユーティリティーを使用します。

1. スーパーユーザーになります。
2. **kclient** インストールスクリプトを実行します。

   インストールには、次の情報が必要です。

   - Kerberos レルム名
   - KDC マスターホスト名
   - 管理主体名
   - 管理主体のパスワード

例 23–8 kclient インストールユーティリティーの実行

次の出力は、kclient コマンドの実行結果を示しています。

```
client# /usr/sbin/kclient

Starting client setup
------------------------------------------------

Do you want to use DNS for kerberos lookups ? [y/n]: n
        No action performed.
Enter the Kerberos realm: EXAMPLE.COM
Specify the KDC hostname for the above realm: kdc1.example.com

Setting up /etc/krb5/krb5.conf.

Enter the krb5 administrative principal to be used: clntconfig/admin
Obtaining TGT for clntconfig/admin ...
Password for clntconfig/admin@EXAMPLE.COM:      <Type the password>
Do you plan on doing Kerberized nfs ? [y/n]: n

host/client.example.com entry ADDED to KDC database.
host/client.example.com entry ADDED to keytab.

Do you want to copy over the master krb5.conf file ? [y/n]: y
Enter the pathname of the file to be copied: \
/net/denver.example.com/export/install/krb5.conf

Copied /net/denver.example.com/export/install/krb5.conf.

------------------------------------------------
Setup COMPLETE !
#
```

▼ Kerberos クライアントを手動で構成する方法

この手順では、次の構成パラメータを使用します。

- レルム名 = EXAMPLE.COM
- DNS ドメイン名 = example.com
- マスター KDC = kdc1.example.com
- スレーブ KDC = kdc2.example.com
- NFS サーバー = denver.example.com
- クライアント = client.example.com
- admin 主体 = kws/admin
- ユーザー主体 = mre

- オンラインヘルプ URL =
  `http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956`

注 – この URL は「Kerberos グラフィカル管理ツール」のセクションを指すように調整してください (428 ページの「Kerberos グラフィカル管理ツールでのオンラインヘルプ URL」を参照)。

---

1 スーパーユーザーになります。

2 **Kerberos 構成ファイル (`krb5.conf`) を編集します。**

デフォルトの Kerberos ファイルを変更する場合は、レルム名とサーバー名を変更する必要があります。`gkadmin` のヘルプファイルへのパスも指定する必要があります。

```
kdc1 # cat /etc/krb5/krb5.conf
[libdefaults]
        default_realm = EXAMPLE.COM

[realms]
        EXAMPLE.COM = {
        kdc = kdc1.example.com
        kdc = kdc2.example.com
        admin_server = kdc1.example.com
        }

[domain_realm]
        .example.com = EXAMPLE.COM
#
# if the domain name and realm name are equivalent,
# this entry is not needed
#
[logging]
        default = FILE:/var/krb5/kdc.log
        kdc = FILE:/var/krb5/kdc.log

[appdefaults]
    gkadmin = {
        help_url = http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956
```

注 – 暗号化タイプを制限する場合は、`default_tkt_enctypes` または `default_tgs_enctypes` の行を設定します。暗号化タイプの制限に関する詳細は、590 ページの「Kerberos 暗号化タイプの使用」を参照してください。

---

3 **(省略可能) KDC の検出に使用されるプロセスを変更します。**

Solaris 10 5/08 リリース以降のデフォルトでは、Kerberos レルムから KDC へのマッピングが決定される順序は次のとおりです。

- `krb5.conf` 内の `realms` セクションの定義。
- DNS 内の SRV レコードの検索による。

krb5.conf ファイルの libdefaults セクションに dns_lookup_kdc または dns_fallback を追加して、この動作を変更できます。詳細は、krb5.conf(4) のマニュアルページを参照してください。常にリフェラルが最初に試行されます。

4 (省略可能) ホストのレルムの決定に使用されるプロセスを変更します。

Solaris 10 5/08 リリース以降のデフォルトでは、ホストからレルムへのマッピングが決定される順序は次のとおりです。

- KDC がリフェラルをサポートしている場合は、KDC からクライアントに、ホストが属しているレルムが通知されることがある。
- krb5.conf ファイル内の domain_realm の定義による。
- ホストの DNS ドメイン名。
- デフォルトレルム。

krb5.conf ファイルの libdefaults セクションに dns_lookup_kdc または dns_fallback を追加して、この動作を変更できます。詳細は、krb5.conf(4) のマニュアルページを参照してください。常にリフェラルが最初に試行されます。

5 (省略可能) NTP などのクロック同期メカニズムを使用して、クライアントのクロックをマスター KDC のクロックと同期化します。

Network Time Protocol (NTP) のインストールと使用は必要はありません。ただし、認証が正常終了するには、すべてのクロックが、krb5.conf ファイル内の clockskew 関係指定子で定義されている最大の誤差以内で KDC サーバー上の時刻と同期化されている必要があります。NTP については、474 ページの「KDC と Kerberos クライアントのクロックの同期化」を参照してください。

6 kadmin を起動します。

Kerberos グラフィカル管理ツールを使って主体を追加する方法については、531 ページの「新しい Kerberos 主体を作成する方法」を参照してください。追加するときは、マスター KDC を構成するときに作成した admin 主体名を使用してログインする必要があります。ただし、次の例では、コマンド行を使用して、必要な主体を追加しています。

```
denver # /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin:
```

a. (省略可能) ユーザー主体が存在しない場合は、ユーザー主体を作成します。

このホストに関連付けられているユーザーに主体が割り当てられていない場合だけ、ユーザー主体を作成します。

```
kadmin: addprinc mre
Enter password for principal mre@EXAMPLE.COM:      <Type the password>
Re-enter password for principal mre@EXAMPLE.COM:      <Type it again>
kadmin:
```

b. **(省略可能) `root` 主体を作成し、その主体をサーバーのキータブファイルに追加します。**

この手順は、クライアントが NFS サービスによってマウントされたファイルシステムに `root` アクセスを持つために必要です。また、cron ジョブを `root` として実行する場合など、非対話的な `root` アクセスが必要な場合にもこの手順が必要になります。

クライアントが NFS サービスを使用してマウントされている遠隔ファイルシステムへの `root` アクセスを必要としない場合は、この手順をスキップできます。`root` 主体は 2 つの構成要素主体とすべきであり、二番目の構成要素は Kerberos クライアントシステムのホスト名にして、レルム幅の `root` 主体の作成を回避します。主体のインスタンスがホスト名のときは、`/etc/resolv.conf` ファイル内のドメイン名が大文字であるか小文字であるかに関係なく、FQDN は小文字で指定する必要があります。

```
kadmin: addprinc -randkey root/client.example.com
Principal "root/client.example.com" created.
kadmin: ktadd root/client.example.com
Entry for principal root/client.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

c. **`host` 主体を作成し、その主体をサーバーのキータブファイルに追加します。**

`host` 主体は、認証を提供するために遠隔アクセスサービスによって使用されます。キータブファイルにまだ資格が存在しない場合は、この主体によって `root` が資格を取得できるようになります。

```
kadmin: addprinc -randkey host/denver.example.com
Principal "host/denver.example.com@EXAMPLE.COM" created.
kadmin: ktadd host/denver.example.com
Entry for principal host/denver.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

d. **(省略可能) サーバーの NFS サービス主体をサーバーのキータブファイルに追加します。**

この手順が必要になるのは、クライアントが Kerberos 認証を使用する NFS ファイルシステムにアクセスする必要がある場合だけです。

```
kadmin: ktadd nfs/denver.example.com
Entry for principal nfs/denver.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

e. **kadmin を終了します。**

```
kadmin: quit
```

7 **(省略可能) NFS での Kerberos を有効にします。**

a. **/etc/nfssec.conf ファイル内の Kerberos セキュリティーモードを有効にします。**

/etc/nfssec.conf ファイルを編集して、Kerberos セキュリティーモードの先頭にある「#」を削除します。

```
# cat /etc/nfssec.conf
 .
 .
#
# Uncomment the following lines to use Kerberos V5 with NFS
#
krb5            390003  kerberos_v5     default -               # RPCSEC_GSS
krb5i           390004  kerberos_v5     default integrity       # RPCSEC_GSS
krb5p           390005  kerberos_v5     default privacy         # RPCSEC_GSS
```

b. **DNS を有効にします。**

/etc/resolv.conf ファイルがまだ作成されていない場合は、このファイルを作成します。サービス主体の正規化は DNS に依存して正規化を実行するためです。詳細は、resolv.conf(4) のマニュアルページを参照してください。

c. **gssd サービスを再開します。**

/etc/resolv.conf ファイルを作成または変更したら、gssd デーモンを再起動して、すべての変更を再読み込みする必要があります。

```
# svcadm restart network/rpc/gss
```

8 クライアントから自動的に **TGT** を更新するか、または **Kerberos** チケットの有効期限切れをユーザーに警告する場合は、**/etc/krb5/warn.conf** ファイルにエントリを作成します。

詳細は、warn.conf(4) のマニュアルページを参照してください。

例 23–9 Solaris 以外の KDC を使用するように Kerberos クライアントを設定する

Solaris 以外の KDC を使用するように Kerberos クライアントを設定することができます。この場合、/etc/krb5/krb5.conf ファイルの realms セクションに、1 行を追加する必要があります。この行を追加すると、クライアントが Kerberos パスワード変更サーバーとの通信に使用するプロトコルが変更されます。この行の書式は次のとおりです。

```
[realms]
                EXAMPLE.COM = {
                kdc = kdc1.example.com
                kdc = kdc2.example.com
                admin_server = kdc1.example.com
                kpasswd_protocol = SET_CHANGE
        }
```

例 23–10 ホスト名とドメイン名を Kerberos レルムにマッピングするための DNS TXT レコード

```
@ IN SOA kdc1.example.com root.kdc1.example.com (
                                1989020501   ;serial
                                10800        ;refresh
                                3600         ;retry
                                3600000      ;expire
                                86400 )      ;minimum

                        IN      NS      kdc1.example.com.
kdc1                    IN      A       192.146.86.20
kdc2                    IN      A       192.146.86.21

_kerberos.example.com.              IN      TXT     "EXAMPLE.COM"
_kerberos.kdc1.example.com.         IN      TXT     "EXAMPLE.COM"
_kerberos.kdc2.example.com.         IN      TXT     "EXAMPLE.COM"
```

例 23–11 Kerberos サーバーの場所を記録する DNS SRV レコード

この例は、KDC、admin サーバー、および kpasswd サーバーの場所を記録するレコードを定義します。

```
@ IN SOA kdc1.example.com root.kdc1.example.com (
                                1989020501   ;serial
                                10800        ;refresh
                                3600         ;retry
                                3600000      ;expire
                                86400 )      ;minimum
```

```
                                IN      NS      kdc1.example.com.
kdc1                            IN      A       192.146.86.20
kdc2                            IN      A       192.146.86.21

_kerberos._udp.EXAMPLE.COM      IN      SRV 0 0 88  kdc2.example.com
_kerberos._tcp.EXAMPLE.COM      IN      SRV 0 0 88  kdc2.example.com
_kerberos._udp.EXAMPLE.COM      IN      SRV 1 0 88  kdc1.example.com
_kerberos._tcp.EXAMPLE.COM      IN      SRV 1 0 88  kdc1.example.com
_kerberos-adm._tcp.EXAMPLE.COM  IN      SRV 0 0 749 kdc1.example.com
_kpasswd._udp.EXAMPLE.COM       IN      SRV 0 0 749 kdc1.example.com
```

▼ チケット認可チケット (TGT) の確認を無効にする方法

この手順では、ローカルの /etc/krb5/krb5.keytab ファイルに保存されているホスト主体の KDC とチケット認可チケットを発行した KDC が同じであることを確認するセキュリティー検査を無効にします。この検査は DNS の偽装攻撃を防止します。ただし、一部のクライアント構成では、ホスト主体を使用できない場合があるため、クライアントが機能できるようにこの検査を無効にする必要があります。次のような構成では、この検査を無効にする必要があります。

- クライアントの IP アドレスが動的に割り当てられる。DHCP クライアントなど。
- クライアントはサービスをホストするように構成されていないため、ホスト主体が作成されていない。
- クライアントにホスト鍵が保存されていない。

1 スーパーユーザーになります。

2 **krb5.conf ファイルを変更します。**

verify_ap_req_nofail オプションが false に設定されている場合、TGT の確認処理は無効になっています。このオプションの詳細については、krb5.conf(4) のマニュアルページを参照してください。

```
client # cat /etc/krb5/krb5.conf
[libdefaults]
        default_realm = EXAMPLE.COM
        verify_ap_req_nofail = false
  ...
```

注 – verify_ap_req_nofail オプションは、krb5.conf ファイルの [libdefaults] セクションまたは [realms] セクションに入力できます。このオプションを [libdefaults] セクションに入力すると、設定はすべてのレルムに使用されます。このオプションを [realms] セクションに入力すると、設定は定義したレルムだけに適用されます。

▼ Kerberos によって保護された NFS ファイルシステムに root ユーザーとしてアクセスする方法

この手順を実行すると、クライアントは root の ID 特権による Kerberos 認証を必要とする、NFS ファイルシステムにアクセスできるようになります。特に、NFS ファイルシステムが次のようなオプションによって共有されている場合です。`-o sec=krb5,root=client1.sun.com`

1 スーパーユーザーになります。

2 **kadmin** を起動します。

Kerberos グラフィカル管理ツールを使って主体を追加する方法については、531 ページの「新しい Kerberos 主体を作成する方法」を参照してください。追加するときは、マスター KDC を構成するときに作成した admin 主体名を使用してログインする必要があります。ただし、次の例では、コマンド行を使用して、必要な主体を追加しています。

```
denver # /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin:
```

a. **NFS クライアントの root 主体を作成します。**

この主体は、Kerberos 認証を必要とする、NFS マウントされたファイルシステムにスーパーユーザーと同様にアクセスするために使用されます。root 主体は 2 つの構成要素主体とすべきであり、二番目の構成要素は Kerberos クライアントシステムのホスト名にして、レルム幅の root 主体の作成を回避します。主体のインスタンスがホスト名のときは、/etc/resolv.conf ファイル内のドメイン名が大文字であるか小文字であるかに関係なく、FQDN は小文字で指定する必要があります。

```
kadmin: addprinc -randkey root/client.example.com
Principal "root/client.example.com" created.
kadmin:
```

b. **サーバーのキータブファイルに root 主体を追加します。**

NFS サービスを使ってマウントされたファイルシステムにクライアントが root アクセスできるように root 主体を追加する場合にこの手順は必要になります。また、cron ジョブを root として実行する場合など、非対話的な root アクセスが必要な場合にもこの手順が必要になります。

```
kadmin: ktadd root/client.example.com
Entry for principal root/client.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type ArcFour
```

```
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

c. **kadmin** を終了します。

```
kadmin: quit
```

▼ Kerberos レルム内のユーザーを自動的に移行するように構成する方法

Kerberos 主体を持たないユーザーを、既存の Kerberos レルムに自動的に移行できます。移行を行うには、/etc/pam.conf のサーバーの認証スタックにある pam_krb5_migrate モジュールの積み重ねにより使用されているサービス用の PAM フレームワークを使用します。

この例では、dtlogin と other の PAM サービス名が構成され、自動移行を使用します。この手順では、次の構成パラメータを使用します。

- レルム名 = EXAMPLE.COM
- マスター KDC = kdc1.example.com
- 移行サービスをホストするマシン = server1.example.com
- 移行サービス主体 = host/server1.example.com

始める前に　レルム EXAMPLE.COM の Kerberos クライアントとして server1 を設定します。詳細は、460 ページの「Kerberos クライアントの構成」を参照してください。

1 **server1** のホストサービス主体が存在するかどうかを確認します。

server1 の keytab ファイル内のホストサービス主体は、マスター KDC にサーバーを認証するために使用されます。

```
server1 # klist -k
Keytab name: FILE:/etc/krb5/krb5.keytab
    KVNO Principal
    ---- --------------------------------------------------
       3 host/server1.example.com@EXAMPLE.COM
       3 host/server1.example.com@EXAMPLE.COM
       3 host/server1.example.com@EXAMPLE.COM
       3 host/server1.example.com@EXAMPLE.COM
```

2 **PAM** 構成ファイルを変更します。

a. **dtlogin** サービスのエントリを追加します。

```
# cat /etc/pam.conf
 .
 .
#
```

```
# dtlogin service (explicit because of pam_krb5_migrate)
#
dtlogin       auth requisite          pam_authtok_get.so.1
dtlogin       auth required           pam_dhkeys.so.1
dtlogin       auth required           pam_unix_cred.so.1
dtlogin       auth sufficient         pam_krb5.so.1
dtlogin       auth requisite          pam_unix_auth.so.1
dtlogin       auth optional           pam_krb5_migrate.so.1
```

b. (省略可能) 必要に応じて、即座のパスワードの変更を強制します。

新しく作成された Kerberos アカウントは、即座の Kerberos パスワードの変更を強制するために、パスワードの有効期限が現在時刻 (今) に設定されています。有効期限を現在時刻に設定するには、`pam_krb5_migrate` モジュールを使用する行に `expire_pw` オプションを追加します。詳細は、`pam_krb5_migrate(5)` のマニュアルページを参照してください。

```
# cat /etc/pam.conf
 .
 .
dtlogin  auth optional           pam_krb5_migrate.so.1 expire_pw
```

c. **pam_krb5** モジュールをアカウントスタックに追加します。

この追加により、Kerberos のパスワードの有効期限でアクセスをブロックできるようになります。

```
# cat /etc/pam.conf
 .
 .
#
# Default definition for Account management
# Used when service name is not explicitly mentioned for account management
#
other   account requisite       pam_roles.so.1
other   account required        pam_krb5.so.1
other   account required        pam_unix_account.so.1
```

d. **pam_krb5** モジュールをパスワードスタックに追加します。

この追加により、パスワードが期限切れになったらパスワードを更新できるようになります。

```
# cat /etc/pam.conf
 .
 .
#
# Default definition for Password management
# Used when service name is not explicitly mentioned for password management
#
other   password required       pam_dhkeys.so.1
other   password requisite      pam_authtok_get.so.1
other   password requisite      pam_authtok_check.so.1
other   password sufficient     pam_krb5.so.1
other   password required       pam_authtok_store.so.1
```

3 マスター KDC 上で、アクセス制御ファイルを更新します。

次のエントリが、root 以外のすべてのユーザーの host/server1.example.com サービス主体に対して、次のエントリは特権の付与、移行、および照会を行います。移行すべきでないユーザーは、kadm5.acl ファイル内で、U 特権を使って示されます。これらのエントリは、permit all または ui エントリの前に置く必要があります。詳細は、kadm5.acl(4) のマニュアルページを参照してください。

```
kdc1 # cat /etc/krb5/kadm5.acl
host/server1.example.com@EXAMPLE.COM U root
host/server1.example.com@EXAMPLE.COM ui *
*/admin@EXAMPLE.COM *
```

4 マスター KDC 上で、Kerberos 管理デーモンを再起動します。

この手順により、kadmind デーモンが新しい kadm5.acl エントリを使用できるようになります。

```
kdc1 # svcadm restart network/security/kadmin
```

5 マスター KDC 上で、pam.conf ファイルにエントリを追加します。

次のエントリにより、kadmind デーモンが k5migrate PAM サービスを使用して、移行に必要なアカウントの UNIX ユーザーパスワードを検査できるようになります。

```
# grep k5migrate /etc/pam.conf
k5migrate       auth    required        pam_unix_auth.so.1
k5migrate       account required        pam_unix_account.so.1
```

KDC と Kerberos クライアントのクロックの同期化

Kerberos 認証システムに参加するすべてのホストは、指定した最大時間内に収まるように内部クロックを同期化する必要があります (「クロックスキュー」)。この必要条件は、Kerberos セキュリティーの検査の 1 つです。参加しているホスト間のクロックスキューが超過すると、クライアントの要求が拒否されます。

アプリケーションサーバーが再実行要求を認識し拒否する目的で、すべての Kerberos プロトコルメッセージをどのくらいの間追跡管理する必要があるかも、クロックスキューで決まります。そのため、クロックスキュー値が長いほど、アプリケーションサーバーが収集する情報も多くなります。

最大クロックスキューのデフォルト値は、300 秒 (5 分) です。このデフォルトは、krb5.conf ファイルの libdefaults セクションで変更できます。

注 – セキュリティー上の理由から、クロックスキュー値は 300 秒より大きくしないでください。

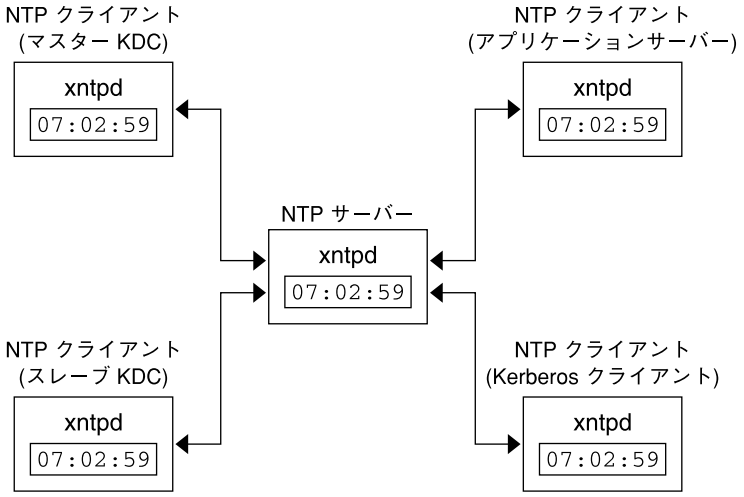
KDC と Kerberos クライアント間で同期化したクロックを管理することは重要であるため、NTP (Network Time Protocol) ソフトウェアを使用して同期化します。Oracle Solaris ソフトウェアにはデラウェア大学の NTP パブリックドメインソフトウェアが含まれています。

注 – クロックを同期化するときは、`rdate` コマンドと `cron` ジョブを使用することもできます。この方法は、NTP より簡単に使用できます。ただし、ここでは NTP を中心に説明します。ネットワークを使用してクロックを同期化する場合は、クロック同期化プロトコル自体も安全である必要があります。

NTP を使用すると、正確な時刻とネットワーククロック同期をネットワーク環境で管理できます。NTP は基本的にはクライアントサーバー実装の状態をとります。1 つのシステムをマスタークロック (NTP サーバー) として指定します。次に、その他のすべてのシステム (NTP クライアント) をマスタークロックと同期するように設定します。

クロックを同期化するために、NTP は `xntpd` デーモンを使用して、インターネット標準時サーバーに合わせて UNIX システムの時刻を設定および管理します。次の図は、NTP のクライアントサーバー実装の例です。

図 **23–1** NTP を使用したクロック同期



KDC および Kerberos クライアントがクロックを同期化するには、次の手順を実行します。

1. ネットワークに NTP サーバーを設定します。NTP サーバーは、マスター KDC 以外であればどのシステムでも設定できます。NTP サーバーの作業については、『Solaris のシステム管理 (ネットワークサービス)』の「NTP の管理 (作業)」を参照してください。
2. ネットワークの KDC と Kerberos クライアントを構成するときに、それらを NTP サーバーの NTP クライアントとして設定します。NTP クライアントの作業については、『Solaris のシステム管理 (ネットワークサービス)』の「NTP の管理 (作業)」を参照してください。

マスター KDC とスレーブ KDC の入れ替え

マスター KDC とスレーブ KDC を入れ替えるときは、この節で説明する手順を行います。マスター KDC とスレーブ KDC の入れ替えは、マスター KDC に何らかの理由で障害が発生した場合、またはマスター KDC を再インストールする必要がある場合 (新しいハードウェアをインストールした場合など) にだけ行ってください。

▼ 入れ替え可能なスレーブ KDC を構成する方法

この手順は、マスター KDC に入れ替え可能なスレーブ KDC に対して実行します。この手順は、増分伝播を使用していることを想定しています。

1 **KDC** をインストールするときに、マスター **KDC** および入れ替え可能なスレーブ **KDC** に対して別名を使用します。

KDC に対してホスト名を定義するときは、各システムの別名が DNS に登録されている必要があります。`/etc/krb5/krb5.conf` ファイルにホストを定義するときも、別名を使用します。

2 手順に従って、スレーブ **KDC** をインストールします。

入れ替えするサーバーは、レルム内でスレーブ KDC として動作している必要があります。手順については、445 ページの「スレーブ KDC を手動で構成する方法」を参照してください。

3 マスター **KDC** コマンドを移動します。

このスレーブ KDC からマスター KDC コマンドが実行されることを防ぐために、`kprop`、`kadmind`、および `kadmin.local` コマンドを別の場所に移動します。

```
kdc4 # mv /usr/lib/krb5/kprop /usr/lib/krb5/kprop.save
kdc4 # mv /usr/lib/krb5/kadmind /usr/lib/krb5/kadmind.save
kdc4 # mv /usr/sbin/kadmin.local /usr/sbin/kadmin.local.save
```

▼ マスター KDC とスレーブ KDC を入れ替える方法

この手順では、旧マスター KDC サーバー名は、kdc1 です。新しいマスター KDC となるスレーブ KDC の名前は、kdc4 です。この手順は、増分伝播を使用していることを想定しています。

始める前に　この手順を実行するには、スレーブ KDC が入れ替え可能なスレーブとして設定されている必要があります。詳細は、476 ページの「入れ替え可能なスレーブ KDC を構成する方法」を参照してください。

1 新しいマスター KDC 上で、**kadmin** を起動します。

```
kdc4 # /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin:
```

a. **kadmind** サービスの新しい主体を作成します。

次の例では、addprinc が 2 行で表示されていますが、1 行に入力する必要があります。

```
kadmin: addprinc -randkey -allow_tgs_req +password_changing_service -clearpolicy \
       changepw/kdc4.example.com
Principal "changepw/kdc4.example.com@ENG.SUN.COM" created.
kadmin: addprinc -randkey -allow_tgs_req -clearpolicy kadmin/kdc4.example.com
Principal "kadmin/kdc4.example.com@EXAMPLE.COM" created.
kadmin:
```

b. キータブファイルを作成します。

```
kadmin: ktadd -k /etc/krb5/kadm5.keytab kadmin/kdc4.example.com
Entry for principal kadmin/kdc4.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc4.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc4.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc4.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc4.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin: ktadd -k /etc/krb5/kadm5.keytab changepw/kdc4.example.com
Entry for principal changepw/kdc4.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc4.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc4.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc4.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc4.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin: ktadd -k /etc/krb5/kadm5.keytab kadmin/changepw
Entry for principal kadmin/changepw with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
```

```
Entry for principal kadmin/changepw with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin:
```

c. **kadmin** を終了します。

```
kadmin: quit
```

2 新しいマスター **KDC** 上で、同期を強制します。

次の手順は、スレーブサーバー上で強制的に KDC を完全に更新します。

```
kdc4 # svcadm disable network/security/krb5kdc
kdc4 # rm /var/krb5/principal.ulog
```

3 新しいマスター **KDC** 上で、更新が完了したことを確認します。

```
kdc4 # /usr/sbin/kproplog -h
```

4 新しいマスター **KDC** 上で、**KDC** サービスを再開します。

```
kdc4 # svcadm enable -r network/security/krb5kdc
```

5 新しいマスター **KDC** 上で、更新ログを消去します。

この手順は、新しいマスター KDC サーバーの更新ログを再度初期化します。

```
kdc4 # svcadm disable network/security/krb5kdc
kdc4 # rm /var/krb5/principal.ulog
```

6 旧マスター **KDC** 上で、**kadmind** プロセスと**krb5kdc** プロセスを終了します。

kadmind プロセスを終了するときは、旧 KDC データベースに対する変更は行わないでください。

```
kdc1 # svcadm disable network/security/kadmin
kdc1 # svcadm disable network/security/krb5kdc
```

7 旧マスター **KDC** 上で、伝播を要求するポーリング時間を指定します。

/etc/krb5/kdc.conf 内の sunw_dbprop_master_ulogsize エントリをコメントにして、sunw_dbprop_slave_poll を定義するエントリを追加します。エントリは、ポーリング時間を 2 分に設定します。

```
kdc1 # cat /etc/krb5/kdc.conf
[kdcdefaults]
        kdc_ports = 88,750

[realms]
        EXAMPLE.COM= {
                profile = /etc/krb5/krb5.conf
                database_name = /var/krb5/principal
                admin_keytab = /etc/krb5/kadm5.keytab
```

```
                acl_file = /etc/krb5/kadm5.acl
                kadmind_port = 749
                max_life = 8h 0m 0s
                max_renewable_life = 7d 0h 0m 0s
                sunw_dbprop_enable = true
#               sunw_dbprop_master_ulogsize = 1000
                sunw_dbprop_slave_poll = 2m
        }
```

8 旧マスター **KDC** 上で、マスター **KDC** コマンドと **kadm5.acl** ファイルを移動します。

マスター KDC コマンドが実行されることを防ぐために、`kprop`、`kadmind`、および `kadmin.local` コマンドを別の場所に移動します。

```
kdc1 # mv /usr/lib/krb5/kprop /usr/lib/krb5/kprop.save
kdc1 # mv /usr/lib/krb5/kadmind /usr/lib/krb5/kadmind.save
kdc1 # mv /usr/sbin/kadmin.local /usr/sbin/kadmin.local.save
kdc1 # mv /etc/krb5/kadm5.acl /etc/krb5/kadm5.acl.save
```

9 **DNS** サーバー上で、マスター **KDC** の別名を変更します。

サーバーを変更するために、`example.com` ゾーンファイルを編集して `masterkdc` のエントリを変更します。

```
masterkdc IN CNAME kdc4
```

10 **DNS** サーバー上で、インターネットドメインネームサーバーを再起動します。

次のコマンドを実行して、新しい別名情報を再ロードします。

```
# svcadm refresh network/dns/server
```

11 新しいマスター **KDC** 上で、マスター **KDC** コマンドとスレーブ **kpropd.acl** ファイルを移動します。

```
kdc4 # mv /usr/lib/krb5/kprop.save /usr/lib/krb5/kprop
kdc4 # mv /usr/lib/krb5/kadmind.save /usr/lib/krb5/kadmind
kdc4 # mv /usr/sbin/kadmin.local.save /usr/sbin/kadmin.local
kdc4 # mv /etc/krb5/kpropd.acl /etc/krb5/kpropd.acl.save
```

12 新しいマスター **KDC** 上で、**Kerberos** アクセス制御リストファイル **(kadm5.acl)** を作成します。

作成された `/etc/krb5/kadm5.acl` ファイルには、KDC を管理できる主体名がすべて含まれている必要があります。このファイルには、増分伝播を要求するすべてのスレーブもリストされている必要があります。詳細は、`kadm5.acl(4)` のマニュアルページを参照してください。

```
kdc4 # cat /etc/krb5/kadm5.acl
kws/admin@EXAMPLE.COM   *
kiprop/kdc1.example.com@EXAMPLE.COM p
```

13 新しいマスター **KDC** 上で、**kdc.conf** ファイル内の更新ログのサイズを指定します。

sunw_dbprop_slave_poll エントリをコメントにして、sunw_dbprop_master_ulogsize を定義するエントリを追加します。エントリは、ログサイズを 1000 エントリに設定します。

```
kdc1 # cat /etc/krb5/kdc.conf
[kdcdefaults]
        kdc_ports = 88,750

[realms]
        EXAMPLE.COM= {
                profile = /etc/krb5/krb5.conf
                database_name = /var/krb5/principal
                admin_keytab = /etc/krb5/kadm5.keytab
                acl_file = /etc/krb5/kadm5.acl
                kadmind_port = 749
                max_life = 8h 0m 0s
                max_renewable_life = 7d 0h 0m 0s
                sunw_dbprop_enable = true
#               sunw_dbprop_slave_poll = 2m
                sunw_dbprop_master_ulogsize = 1000
        }
```

14 新しいマスター **KDC** 上で、**kiprop** 主体を **kadmind** キータブファイルに追加します。

```
kdc4 # kadmin.local
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kiprop/kdc4.example.com
Entry for principal kiprop/kdc4.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc4.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc4.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc4.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc4.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: quit
```

15 新しいマスター **KDC** 上で、**kadmind** と **krb5kdc** を実行します。

```
kdc4 # svcadm enable -r network/security/krb5kdc
kdc4 # svcadm enable -r network/security/kadmin
```

16 旧マスター **KDC** 上で、**kiprop** サービス主体を追加します。

krb5.keytab ファイルに kiprop 主体を追加すると、増分伝播サービスに対して kpropd デーモンが自身を認証できるようになります。

```
kdc1 # /usr/sbin/kadmin -p kws/admin
Authenticating as pricipal kws/admin@EXAMPLE.COM with password.
Enter password:     <Type kws/admin password>
kadmin: ktadd kiprop/kdc1.example.com
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
```

```
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

17 旧マスター **KDC** 上で、**krb5.conf** にリストされた各 **KDC** のエントリを伝播構成ファイル **kropd.acl** に追加します。

```
kdc1 # cat /etc/krb5/kropd.acl
host/kdc1.example.com@EXAMPLE.COM
host/kdc2.example.com@EXAMPLE.COM
host/kdc3.example.com@EXAMPLE.COM
host/kdc4.example.com@EXAMPLE.COM
```

18 旧マスター **KDC** 上で、**kpropd** と **krb5kdc** を実行します。

krb5kdc デーモンが起動すると、システムがスレーブとして構成されている場合、kpropd も起動します。

```
kdc1 # svcadm enable network/security/krb5kdc
```

Kerberos データベースの管理

Kerberos データベースは、Kerberos の最も重要な構成要素であるため、適切に管理する必要があります。この節では、データベースのバックアップと復元、増分または並列伝播の設定、stash ファイルの管理など、Kerberos データベースの管理についていくつかの手順を説明します。データベースを初期設定する手順については、431 ページの「マスター KDC を手動で構成する方法」を参照してください。

Kerberos データベースのバックアップと伝播

マスター KDC の Kerberos データベースをスレーブ KDC に伝播する処理は、構成処理の中で最も重要なものの 1 つです。伝播の頻度が低いと、マスター KDC とスレーブ KDC が同期しなくなります。マスター KDC に障害が発生した場合、スレーブ KDC は最新のデータベース情報を持たないことになります。また、負荷を分散するためにスレーブ KDC がマスター KDC として構成されている場合も、そのスレーブ KDC をマスター KDC として使用するクライアントは最新情報を取得できません。このため、Kerberos データベースの変更頻度に基づいて、伝播頻度を適切に設定する必要があります。増分伝播は手動による伝播より優先されます。これは、データベースを手動で伝播すると管理のオーバーヘッドがより大きくなるためです。また、データベースを完全に伝播する場合は、効率が悪いためです。

マスター KDC を構成するときは、cron ジョブ内に kprop_script コマンドを設定して、Kerberos データベースを /var/krb5/slave_datatrans ダンプファイルに自動的にバックアップし、それをスレーブ KDC に伝播します。ただし、他のファイルと同様

に、Kerberos データベースは壊れることがあります。スレーブ KDC のデータが壊れた場合でも、次回のデータベース自動伝播によって最新のコピーがインストールされるため、影響が発生しないこともあります。ただし、マスター KDC のデータが壊れた場合は、壊れたデータベースが次回の伝播ですべてのスレーブ KDC に伝播されます。また、壊れたデータがバックアップされると、マスター KDC 上の壊れていない前回のバックアップファイルが上書きされます。

この場合、安全なバックアップコピーが存在しないため、`cron` ジョブを設定して `slave_datatrans` ダンプファイルを定期的に別の場所にコピーするか、`kdb5_util` の `dump` コマンドを使用して別のバックアップコピーを作成することも必要です。これにより、データベースが壊れても、`kdb5_util` の `load` コマンドを使用して、マスター KDC の最新のバックアップを復元することができます。

次の点も重要です。 データベースダンプファイルには主体鍵が含まれているため、許可されないユーザーがアクセスできないように、ファイルを保護する必要があります。デフォルトでは、データベースダンプファイルの読み取り権および書き込み権は、`root` にだけ与えられます。許可されないアクセスから保護するには、`kprop` コマンドだけを使用して、データベースダンプファイルを伝播します。この場合、転送するデータが暗号化されます。また、`kprop` はデータをスレーブ KDC だけに伝播するため、データベースダンプファイルが間違って許可されないホストに送信される可能性が最小限になります。

注意 – Kerberos データベースが伝播されたあとに更新され、次の伝播の前にデータベースが壊れた場合は、スレーブ KDC には更新が反映されません。この更新は失われます。このため、定期的に実行される伝播の前に重要な更新を追加した場合は、データの損失を回避するために手動でデータベースを伝播する必要があります。

kpropd.acl ファイル

スレーブ KDC の `kpropd.acl` ファイルの各行には、ホスト主体名と、伝播された最新のデータベースの受信元となるシステムが指定されています。マスター KDC を使用してすべてのスレーブ KDC に伝播する場合は、各スレーブ KDC の `kpropd.acl` ファイルに対してマスター KDC の主体名だけを指定する必要があります。

ただし、このドキュメントの Kerberos のインストールおよびインストール後の構成手順では、マスター KDC とスレーブ KDC に対して同じ `kpropd.acl` ファイルを追加するように説明しています。このファイルには、すべての KDC ホスト主体名が含まれます。この構成を使用すると、伝播元の KDC が一時的に使用できなくなったときでも、任意の KDC から伝播することができます。また、すべての KDC に同一のコピーを保持すると、構成の管理が容易になります。

kprop_script コマンド

kprop_script コマンドは、kprop コマンドを使用して Kerberos データベースをほかの KDC に伝播します。kprop_script コマンドをスレーブ KDC 上で実行すると、そのスレーブ KDC の Kerberos データベースのコピーがほかの KDC に伝播されます。kprop_script には、ホスト名のリストを引数として指定します。区切り文字は空白です。指定したホスト名は、伝播先の KDC になります。

kprop_script を実行すると、Kerberos データベースのバックアップが /var/krb5/slave_datatrans ファイルに作成され、指定した KDC にそのファイルがコピーされます。Kerberos データベースは、伝播が完了するまでロックされます。

▼ Kerberos データベースをバックアップする方法

1. マスター KDC 上でスーパーユーザーになります。
2. **kdb5_util** の **dump** コマンドを使用して、**Kerberos** データベースをバックアップします。

   # /usr/sbin/kdb5_util dump [-verbose] [-d *dbname*] [*filename* [*principals*...]]

   -verbose　バックアップする各主体とポリシー名を出力します。

   dbname　バックアップするデータベース名を定義します。ファイルの絶対パスを指定できます。-d オプションを指定しない場合、デフォルトのデータベース名は /var/krb5/principal となります。

   filename　データベースのバックアップに使用するファイルを定義します。ファイルの絶対パスを指定できます。ファイルを指定しなかった場合、データベースは標準出力にダンプされます。

   principals　バックアップする主体を 1 つ以上定義します (区切り文字は空白)。主体名は完全指定形式にする必要があります。主体を指定しなかった場合、データベース全体がバックアップされます。

例 23–12 Kerberos データベースのバックアップ

次の例では、Kerberos データベースは dumpfile と呼ばれるファイルにバックアップされます。-verbose オプションが指定されているため、各主体はバックアップされるときに出力されます。

```
# kdb5_util dump -verbose dumpfile
kadmin/kdc1.eng.example.com@ENG.EXAMPLE.COM
krbtgt/ENG.EXAMPLE.COM@ENG.EXAMPLE.COM
kadmin/history@ENG.EXAMPLE.COM
pak/admin@ENG.EXAMPLE.COM
pak@ENG.EXAMPLE.COM
changepw/kdc1.eng.example.com@ENG.EXAMPLE.COM
```

次の例では、pak および pak/admin 主体が、Kerberos データベースからバックアップされます。

```
# kdb5_util dump -verbose dumpfile pak/admin@ENG.EXAMPLE.COM pak@ENG.EXAMPLE.COM
pak/admin@ENG.EXAMPLE.COM
pak@ENG.EXAMPLE.COM
```

▼ Kerberos データベースを復元する方法

1. マスター KDC 上でスーパーユーザーになります。
2. マスター上で、KDC デーモンを終了します。

   ```
   kdc1 # svcadm disable network/security/krb5kdc
   kdc1 # svcadm disable network/security/kadmin
   ```

3. **kdb_util** コマンドの **load** コマンドを使用して、Kerberos データベースを復元します。

   ```
   # /usr/sbin/kdb5_util load [-verbose] [-d dbname] [-update] [filename]
   ```

   -verbose　復元する各主体とポリシー名を出力します。

   dbname　復元するデータベース名を定義します。ファイルの絶対パスを指定できます。-d オプションを指定しない場合、デフォルトのデータベース名は /var/krb5/principal となります。

   -update　既存のデータベースを更新します。指定しない場合、新しいデータベースが作成されるか、既存のデータベースが上書きされます。

   filename　データベースの復元に使用するファイルを定義します。ファイルの絶対パスを指定できます。

4. KDC デーモンを起動します。

   ```
   kdc1 # svcadm enable -r network/security/krb5kdc
   kdc1 # svcadm enable -r network/security/kadmin
   ```

例 23–13 Kerberos データベースの復元

次の例では、database1 というデータベースが、dumpfile ファイルから現在のディレクトリに復元されます。-update オプションが指定されていないため、復元によって新しいデータベースが作成されます。

```
# kdb5_util load -d database1 dumpfile
```

▼ サーバーのアップグレード後に Kerberos データベースを変換する方法

Solaris 8 または Solaris 9 リリースが稼働するサーバー上で KDC データベースが作成されている場合、データベースを変換すると、改善されたデータベースのフォーマットを利用することができます。

始める前に　データベースが古いフォーマットを使用していることを確認してください。

1. マスター上で、**KDC** デーモンを終了します。

```
kdc1 # svcadm disable network/security/krb5kdc
kdc1 # svcadm disable network/security/kadmin
```

2. データベースの一時的なコピーを格納するためのディレクトリを作成します。

```
kdc1 # mkdir /var/krb5/tmp
kdc1 # chmod 700 /var/krb5/tmp
```

3. **KDC** データベースをダンプします。

```
kdc1 # kdb5_util dump /var/krb5/tmp/prdb.txt
```

4. 現在のデータベースファイルのコピーを保存します。

```
kdc1 # cd /var/krb5
kdc1 # mv princ* tmp/
```

5. データベースをロードします。

```
kdc1 # kdb5_util load /var/krb5/tmp/prdb.txt
```

6. **KDC** デーモンを起動します。

```
kdc1 # svcadm enable -r network/security/krb5kdc
kdc1 # svcadm enable -r network/security/kadmin
```

▼ マスター KDC を再構成して増分伝播を使用する方法

この手順を使って、増分伝播を使用するように既存のマスター KDC を再構成します。この手順では、次の構成パラメータを使用します。

- レルム名 = `EXAMPLE.COM`
- DNS ドメイン名 = `example.com`
- マスター KDC = `kdc1.example.com`
- スレーブ KDC = `kdc2.example.com`
- `admin` 主体 = `kws/admin`

1 **kdc.conf** にエントリを追加します。

増分伝播を有効にして、ログ内に保持する KDC マスターの更新数を選択する必要があります。詳細は、kdc.conf(4) のマニュアルページを参照してください。

```
kdc1 # cat /etc/krb5/kdc.conf
[kdcdefaults]
        kdc_ports = 88,750

[realms]
        EXAMPLE.COM= {
                profile = /etc/krb5/krb5.conf
                database_name = /var/krb5/principal
                admin_keytab = /etc/krb5/kadm5.keytab
                acl_file = /etc/krb5/kadm5.acl
                kadmind_port = 749
                max_life = 8h 0m 0s
                max_renewable_life = 7d 0h 0m 0s
                sunw_dbprop_enable = true
                sunw_dbprop_master_ulogsize = 1000
        }
```

2 **kiprop** 主体を作成します。

kiprop 主体は、マスター KDC サーバーの認証とマスター KDC からの更新の認証に使用されます。

```
kdc1 # /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin: addprinc -randkey kiprop/kdc1.example.com
Principal "kiprop/kdc1.example.com@EXAMPLE.COM" created.
kadmin: addprinc -randkey kiprop/kdc2.example.com
Principal "kiprop/kdc2.example.com@EXAMPLE.COM" created.
kadmin:
```

3 **kadmind** キータブファイルに **kiprop** 主体を追加します。

kadm5.keytab に kiprop 主体を追加すると、起動時に kadmind コマンドが自身を認証できます。

```
kadmin: ktadd -k /etc/krb5/kadm5.keytab kiprop/kdc1.example.com
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin: quit
```

4 マスター **KDC** 上で、**kadm5.acl** に **kiprop** エントリを追加します。

このエントリにより、マスター KDC が kdc2 サーバーから増分伝播の要求を受け取ることができるようになります。

```
kdc1 # cat /etc/krb5/kadm5.acl
*/admin@EXAMPLE.COM *
kiprop/kdc2.example.com@EXAMPLE.COM p
```

5 **root** の **crontab** ファイルの **kprop** 行をコメントにします。

この手順により、マスター KDC が KDC データベースのコピーを伝播しなくなります。

```
kdc1 # crontab -e
#ident  "@(#)root       1.20    01/11/06 SMI"
#
# The root crontab should be used to perform accounting data collection.
#
# The rtc command is run to adjust the real time clock if and when
# daylight savings time changes.
#
10 3 * * * /usr/sbin/logadm
15 3 * * 0 /usr/lib/fs/nfs/nfsfind
1 2 * * * [ -x /usr/sbin/rtc ] && /usr/sbin/rtc -c > /dev/null 2>&1
30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
#10 3 * * * /usr/lib/krb5kprop_script kdc2.example.sun.com #SUNWkr5ma
```

6 **kadmind** を再起動します。

```
kdc1 # svcadm restart network/security/kadmin
```

7 増分伝播を使用するすべてのスレーブ **KDC** サーバーを再構成します。

詳細な手順については、487 ページの「スレーブ KDC を再構成して増分伝播を使用する方法」を参照してください。

▼ スレーブ KDC を再構成して増分伝播を使用する方法

1 **krb5.conf** にエントリを追加します。

新しいエントリにより、増分伝播が有効にされ、ポーリング時間が 2 分に設定されます。

```
kdc2 # cat /etc/krb5/krb5.conf
[kdcdefaults]
        kdc_ports = 88,750

[realms]
        EXAMPLE.COM= {
                profile = /etc/krb5/krb5.conf
                database_name = /var/krb5/principal
                admin_keytab = /etc/krb5/kadm5.keytab
```

```
                        acl_file = /etc/krb5/kadm5.acl
                        kadmind_port = 749
                        max_life = 8h 0m 0s
                        max_renewable_life = 7d 0h 0m 0s
                        sunw_dbprop_enable = true
                        sunw_dbprop_slave_poll = 2m
                }
```

2 **krb5.keytab** ファイルに **kiprop** 主体を追加します。

```
kdc2 # /usr/sbin/kadmin -p kws/admin
Enter password:     <Type kws/admin password>
kadmin: ktadd kiprop/kdc2.example.com
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

3 **kpropd** を無効にします。

```
kdc2 # svcadm disable network/security/krb5_prop
```

4 **KDC** サーバーを再起動します。

```
kdc2 # svcadm restart network/security/krb5kdc
```

▼ 完全伝播を使用するようにスレーブ KDC を構成する方法

この手順は Solaris 10 リリースを実行しているスレーブ KDC サーバーを再構成して完全伝播を使用する方法を示しています。通常、この手順は、マスター KDC サーバーが Solaris 9 リリースまたはそれ以前のリリースのいずれかを実行している場合にのみ使用する必要があります。この場合、マスター KDC サーバーは増分伝播をサポートできないので、伝播が機能するようにスレーブを構成する必要があります。

この手順では、`kdc3` という名前のスレーブ KDC を構成します。この手順では、次の構成パラメータを使用します。

- レルム名 = `EXAMPLE.COM`
- DNS ドメイン名 = `example.com`

- マスター KDC = kdc1.example.com
- スレーブ KDC = kdc2.example.com and kdc3.example.com
- admin 主体 = kws/admin
- オンラインヘルプ URL = http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956

注 – この URL は「Kerberos グラフィカル管理ツール」のセクションを指すように調整してください (428 ページの「Kerberos グラフィカル管理ツールでのオンラインヘルプ URL」を参照)。

始める前に　マスター KDC が構成済みである必要があります。スレーブ KDC を入れ替え可能にする手順については、476 ページの「マスター KDC とスレーブ KDC の入れ替え」を参照してください。

1. マスター **KDC** 上でスーパーユーザーになります。

2. マスター **KDC** 上で **kadmin** を起動します。

   マスター KDC を構成するときに作成した admin 主体名を使用して、ログインする必要があります。

   ```
   kdc1 # /usr/sbin/kadmin -p kws/admin
   Enter password:      <Type kws/admin password>
   kadmin:
   ```

   a. マスター **KDC** のデータベースにスレーブホスト主体が存在しない場合は、追加します。

      スレーブが機能するには、ホスト主体が必要です。主体のインスタンスがホスト名のときは、/etc/resolv.conf ファイル内のドメイン名が大文字であるか小文字であるかに関係なく、FQDN は小文字で指定する必要があります。

      ```
      kadmin: addprinc -randkey host/kdc3.example.com
      Principal "host/kdc3@EXAMPLE.COM" created.
      kadmin:
      ```

   b. **kadmin** を終了します。

      ```
      kadmin: quit
      ```

3. マスター **KDC** 上で、**Kerberos** 構成ファイル (**krb5.conf**) を編集します。

   各スレーブ用にエントリを追加する必要があります。このファイルの詳細は、krb5.conf(4) のマニュアルページを参照してください。

   ```
   kdc1 # cat /etc/krb5/krb5.conf
    .
    .
   [realms]
   ```

```
                EXAMPLE.COM = {
                kdc = kdc1.example.com
                kdc = kdc2.example.com
                kdc = kdc3.example.com
                admin_server = kdc1.example.com
        }
```

4 マスター KDC 上で、マスター KDC および各スレーブ KDC のエントリを **kpropd.acl** ファイルに追加します。

このファイルの詳細は、kprop(1M) のマニュアルページを参照してください。

```
kdc1 # cat /etc/krb5/kpropd.acl
host/kdc1.example.com@EXAMPLE.COM
host/kdc2.example.com@EXAMPLE.COM
host/kdc3.example.com@EXAMPLE.COM
```

5 すべてのスレーブ KDC 上で、KDC 管理ファイルをマスター KDC サーバーからコピーします。

この手順は、マスター KDC サーバーが、各 KDC サーバーに必要な情報を更新したため、すべてのスレーブ KDC 上で実行する必要があります。ftp などの転送メカニズムを使用して、マスター KDC から次のファイルのコピーを取得できます。

- /etc/krb5/krb5.conf
- /etc/krb5/kdc.conf
- /etc/krb5/kpropd.acl

6 すべてのスレーブ KDC 上で、Kerberos アクセス制御リストファイル **kadm5.acl** が反映されていないことを確認してください。

修正前の kadm5.acl ファイルは次のようになっています。

```
kdc2 # cat /etc/krb5/kadm5.acl
*/admin@___default_realm___ *
```

ファイルに kiprop のエントリがある場合は、それを削除します。

7 新しいスレーブ上で、**kadmin** コマンドを起動します。

マスター KDC を構成するときに作成した admin 主体名を使用して、ログインする必要があります。

```
kdc2 # /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin:
```

a. **kadmin** を使用して、スレーブの **host** 主体をスレーブのキータブファイルに追加します。

このエントリにより kprop などの Kerberos アプリケーションが機能します。主体のインスタンスがホスト名のときは、/etc/resolv.conf ファイル内のドメイン名が大文字であるか小文字であるかに関係なく、FQDN は小文字で指定する必要があります。

```
kadmin: ktadd host/kdc3.example.com
Entry for principal host/kdc3.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc3.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc3.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc3.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc3.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

b. **kadmin** を終了します。

```
kadmin: quit
```

8 マスター **KDC** 上で、**crontab -e** を実行し、自動的にバックアップを実行する **cron** ジョブにスレーブ **KDC** 名を追加します。そのジョブは、**crontab -e** によって自動的にバックアップを実行します。

各スレーブ KDC サーバーの名前を kprop_script 行の末尾に追加します。

```
10 3 * * * /usr/lib/krb5/kprop_script kdc2.example.com kdc3.example.com
```

バックアップの時刻を変更する場合もあるでしょう。このエントリは、バックアップ処理を毎日午前 3:10 に開始します。

9 新しいスレーブ上で、**Kerberos** 伝播デーモンを起動します。

```
kdc3 # svcadm enable network/security/krb5_prop
```

10 マスター **KDC** 上で、**kprop_script** を使ってデータベースをバックアップし、伝播します。

データベースのバックアップコピーがすでに使用可能な場合は、別のバックアップを完成させる必要はありません。詳細は、493 ページの「Kerberos データベースをスレーブ KDC に手動で伝播する方法」を参照してください。

```
kdc1 # /usr/lib/krb5/kprop_script kdc3.example.com
Database propagation to kdc3.example.com: SUCCEEDED
```

11 新しいスレーブ上で、**kdb5_util** を使用して **stash** ファイルを作成します。

```
kdc3 # /usr/sbin/kdb5_util stash
kdb5_util: Cannot find/read stored master key while reading master key
kdb5_util: Warning: proceeding without master key

Enter KDC database master key:      <Type the key>
```

12 **(省略可能)** 新しいスレーブ **KDC** 上で、**NTP** などのクロック同期メカニズムを使用して、マスター **KDC** のクロックを同期化します。

Network Time Protocol (NTP) のインストールと使用は必要はありません。ただし、認証が正常終了するには、`krb5.conf` ファイルの `libdefaults` セクションに定義されているデフォルト時間内に収まるよう、すべてのクロックを調整する必要があります。NTP については、474 ページの「KDC と Kerberos クライアントのクロックの同期化」を参照してください。

13 新しいスレーブ上で、**KDC** デーモン **(krb5kdc)** を起動します。

```
kdc3 # svcadm enable network/security/krb5kdc
```

▼ KDC サーバーが同期しているかを検査する方法

増分伝播が構成されている場合、この手順は、スレーブ KDC 上の情報が更新されたかを確認します。

1 **KDC** マスターサーバー上で、**kproplog** コマンドを実行します。

```
kdc1 # /usr/sbin/kproplog -h
```

2 **KDC** スレーブサーバー上で、**kproplog** コマンドを実行します。

```
kdc2 # /usr/sbin/kproplog -h
```

3 最終シリアル番号と最終時刻表示の値が一致するかを確認します。

例 23–14 KDC サーバーが同期しているかを検査する

次の例は、マスター KDC サーバー上での `kproplog` コマンドの実行結果です。

```
kdc1 # /usr/sbin/kproplog -h

Kerberos update log (/var/krb5/principal.ulog)
Update log dump:
    Log version #: 1
    Log state: Stable
    Entry block size: 2048
    Number of entries: 2500
    First serial #: 137966
    Last serial #: 140465
    First time stamp: Fri Nov 28 00:59:27 2004
    Last time stamp: Fri Nov 28 01:06:13 2004
```

次の例は、スレーブ KDC サーバー上での kproplog コマンドの実行結果です。

```
kdc2 # /usr/sbin/kproplog -h

Kerberos update log (/var/krb5/principal.ulog)
Update log dump:
    Log version #: 1
    Log state: Stable
    Entry block size: 2048
    Number of entries: 0
    First serial #: None
    Last serial #: 140465
    First time stamp: None
    Last time stamp: Fri Nov 28 01:06:13 2004
```

最終シリアル番号と最終時刻表示が同じであることに注意してください。これは、スレーブがマスター KDC サーバーと同期していることを示しています。

スレーブ KDC サーバーの出力で、スレーブ KDC サーバーの更新ログに更新エントリがないことに注意してください。エントリがないのは、スレーブ KDC サーバーはマスター KDC サーバーとは異なり、一連の更新を保持しないためです。また、最初のシリアル番号と最初の時刻表示は関連情報でないため、KDC スレーブサーバーはそれらの情報を取り込みません。

▼ Kerberos データベースをスレーブ KDC に手動で伝播する方法

この手順では、kprop コマンドを使用して、Kerberos データベースを伝播します。定期的に実行する cron ジョブ以外に、スレーブ KDC とマスター KDC を同期化する必要がある場合は、この手順を行います。kprop_script と異なり、kprop を使用した場合は、現在のデータベースバックアップだけを伝播できます。伝播する前に、Kerberos データベースの新しいバックアップは作成されません。

注 – 増分伝播を使用する場合は、この手順を使用しないでください。

---

1. マスター **KDC** 上でスーパーユーザーになります。
2. **(省略可能) kdb5_util** コマンドを使用して、データベースをバックアップします。

   ```
   # /usr/sbin/kdb5_util dump /var/krb5/slave_datatrans
   ```

3. **kprop** コマンドを使用して、データベースをスレーブ **KDC** に伝播します。

   ```
   # /usr/lib/krb5/kprop -f /var/krb5/slave_datatrans slave-KDC
   ```

例 **23–15** kprop_script を使用してスレーブ KDC に Kerberos データベースを手動で伝播する

定期的に実行する cron ジョブ以外に、データベースをバックアップし、そのファイルをスレーブ KDC に伝播する場合は、次のように kprop_script コマンドを使用することもできます。

```
# /usr/lib/krb5/kprop_script slave-KDC
```

並列伝播の設定

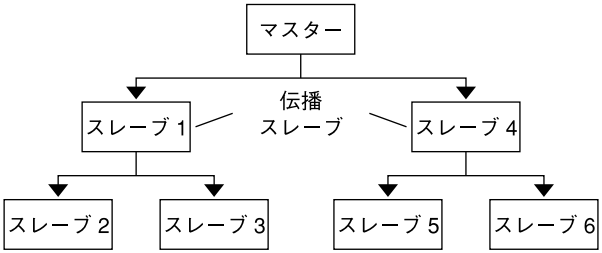
ほとんどの場合、マスター KDC は、Kerberos データベースをスレーブ KDC に伝播するときにだけ使用されます。使用するサイトに複数のスレーブ KDC が存在する場合は、伝播処理の負荷を分散させることもできます。この概念は、「並列伝播」と呼ばれます。

注 – 増分伝播を使用する場合は、この手順を使用しないでください。

並列伝播を利用すると、複数のスレーブ KDC 間でマスター KDC の伝播処理を分散できます。処理を分散すると、伝播をより早く実行でき、マスター KDC の作業を軽減することができます。

たとえば、使用するサイトに 1 つのマスター KDC と 6 つのスレーブ KDC があるとします (図 23–2 を参照)。slave-1 から slave-3 で 1 つの論理グループを構成し、slave-4 から slave-6 で別の論理グループを構成しています。並列伝播を設定するには、マスター KDC がデータベースを slave-1 と slave-4 に伝播し、これらのスレーブ KDC がグループ内のスレーブ KDC にデータベースを伝播するようにします。

図 23–2 並列伝播の構成例



並列伝播を設定するための構成手順

ここでは、並列伝播の詳細な手順は説明しませんが、並列伝播を有効にする構成手順の概要を示します。手順は次のとおりです。

1. マスター KDC 上で、cron ジョブ内の kprop_script エントリを変更して、次の伝播先のスレーブ KDC (伝播スレーブ) だけを引数に指定します。
2. 伝播スレーブごとに、kprop_script エントリをその cron ジョブに追加し、伝播先のスレーブを引数に指定します。並列伝播を正しく行うには、伝播スレーブが新しい Kerberos データベースから伝播されたあとに、cron ジョブが実行されるように設定する必要があります。

注 – 伝播スレーブにかかる伝播時間は、ネットワークの帯域幅や Kerberos データベースのサイズなどの要因によって異なります。

---

3. スレーブ KDC ごとに、伝播に必要なアクセス権を設定します。伝播元の KDC のホスト主体名を各スレーブ KDC の kpropd.acl ファイルに追加します。

例 23–16 並列伝播の設定

図 23–2 の例を使用すると、マスター KDC の kprop_script エントリは次のようになります。

```
0 3 * * * /usr/lib/krb5/kprop_script slave-1.example.com slave-4.example.com
```

slave-1 の kprop_script エントリは、次のようになります。

```
0 4 * * * /usr/lib/krb5/kprop_script slave-2.example.com slave-3.example.com
```

このスレーブの伝播は、マスターからの伝播が完了してから 1 時間後に開始します。

伝播スレーブの kpropd.acl ファイルには、次のエントリが含まれます。

例 23–16 並列伝播の設定 *(続き)*

```
host/master.example.com@EXAMPLE.COM
```

slave-1 から伝播されるスレーブ KDC の kpropd.acl ファイルには、次のエントリが含まれます。

```
host/slave-1.example.com@EXAMPLE.COM
```

stash ファイルの管理

「*stash* ファイル」には、Kerberos データベースのマスター鍵が含まれます。このファイルは、Kerberos データベースを作成すると自動的に作成されます。stash ファイルが壊れた場合は、kdb5_util ユーティリティーの stash コマンドを使用して、置き換えることができます。kdb5_util の destroy コマンドを使用して Kerberos データベースを削除したときは、stash ファイルも削除する必要があります。データベースを削除しても、stash ファイルは自動的に削除されないため、クリーンアップを完了するには、stash ファイルを削除する必要があります。

▼ stash ファイルを削除する方法

1 **stash ファイルが配置されている KDC の上でスーパーユーザーになります。**

2 **stash ファイルを削除します。**

```
# rm stash-file
```

この例では、*stash-file* は stash ファイルのパスを示します。デフォルトでは、stash ファイルは /var/krb5/.k5.*realm* にあります。

注 – stash ファイルを再作成する場合は、kdb5_util コマンドの -f オプションを使用します。

LDAP ディレクトリサーバーでの KDC の管理

LDAP ディレクトリサーバーを使用した KDC 管理作業のほとんどは、DB2 サーバーを使用した場合と同じです。LDAP を使用した処理に特有の新しい作業がいくつかあります。

表 23–3 LDAP を使用するための KDC サーバーの構成 (作業マップ)

タスク	説明	参照先
マスター KDC を構成します。	手動のプロセスを使用し、さらに KDC 用に LDAP を使用して、レルムにマスター KDC サーバーとデータベースを構成および構築します。	437 ページの「LDAP データサーバーを使用するように KDC を構成する方法」
Kerberos 主体属性を Kerberos 以外のオブジェクトクラス型と結び付けます	Kerberos レコードで格納された情報をほかの LDAP データベースと共有できるようになります。	497 ページの「Kerberos 主体属性を Kerberos 以外のオブジェクトクラス型に結び付ける方法」
レルムを破棄します。	レルムに関連付けられたデータをすべて削除します。	498 ページの「LDAP ディレクトリサーバーでレルムを破棄する方法」

▼ Kerberos 主体属性を Kerberos 以外のオブジェクトクラス型に結び付ける方法

この手順により、Kerberos 主体属性を Kerberos 以外のオブジェクトクラス型に関連付けることができます。この手順では、`krbprincipalaux`、`krbTicketPolicyAux`、および `krbPrincipalName` 属性が people オブジェクトクラスに関連付けられます。

この手順では、次の構成パラメータを使用します。

- ディレクトリサーバー = `dsserver.example.com`
- ユーザー主体 = `willf@EXAMPLE.COM`

1 スーパーユーザーになります。

2 **people** オブジェクトクラスの各エントリを用意します。

エントリごとに次の手順を繰り返します。

```
cat << EOF | ldapmodify -h dsserver.example.com -D "cn=directory manager"
dn: uid=willf,ou=people,dc=example,dc=com
changetype: modify
objectClass: krbprincipalaux
objectClass: krbTicketPolicyAux
krbPrincipalName: willf@EXAMPLE.COM
EOF
```

3 サブツリー属性をレルムコンテナに追加します。

この手順により、デフォルトの EXAMPLE.COM コンテナだけでなく、`ou=people,dc=example,dc=com` コンテナでも主体エントリを検索できるようになります。

```
# kdb5_ldap_util -D "cn=directory manager" modify \
            -subtrees 'ou=people,dc=example,dc=com' -r EXAMPLE.COM
```

4 (省略可能) KDC レコードが DB2 に格納されている場合は、DB2 エントリを移行します。

a. DB2 エントリをダンプします。

```
# kdb5_util dump > dumpfile
```

b. データベースを LDAP サーバーにロードします。

```
# kdb5_util load -update dumpfile
```

5 (省略可能) 主体属性を KDC に追加します。

```
# kadmin.local -q 'addprinc willf'
```

▼ LDAP ディレクトリサーバーでレルムを破棄する方法

この手順は、別の LDAP ディレクトリサーバーがレルムを処理するように構成されている場合に使用できます。

1 スーパーユーザーになります。

2 レルムを破棄します。

```
# kdb5_ldap_util -D "cn=directory manager" destroy
```

Kerberos サーバー上のセキュリティーの強化

Kerberos アプリケーションサーバーと KDC サーバーのセキュリティーを強化するには、次の手順に従ってください。

表 23–4 Kerberos サーバーのセキュリティーの強化 (作業マップ)

タスク	説明	説明
Kerberos 認証を使用してアクセスを有効にします。	ネットワークアクセスを制限し、サーバーで Kerberos 認証のみが許可されるようにします。	499 ページの「Kerberos アプリケーションのみを有効にする方法」
KDC サーバーへのアクセスを制限します。	KDC サーバーとそのデータのセキュリティーを強化します。	499 ページの「KDC サーバーへのアクセスを制限する方法」
辞書ファイルを使用してパスワードセキュリティーを強化します。	新しいパスワードを辞書と照合してパスワードのセキュリティーを強化します。	500 ページの「辞書ファイルを使用してパスワードセキュリティーを強化する方法」

▼ Kerberos アプリケーションのみを有効にする方法

この手順を実行すると、telnet、 ftp、 rcp、 rsh、 および rlogin を実行しているサーバーへのネットワークアクセスが、Kerberos 認証されたトランザクションだけを使用するネットワークアクセスに制限されます。

1 **telnet サービスの exec プロパティーを変更します。**

telnet の exec プロパティーに -a user オプションを追加すると、有効な認証情報を提供できるユーザーにアクセスが制限されます。

```
# inetadm -m svc:/network/telnet:default exec="/usr/sbin/in.telnetd -a user"
```

2 **(省略可能) まだ構成されていない場合は、telnet サービスの exec のプロパティーを変更します。**

ftp の exec プロパティーに -a オプションを追加すると、Kerberos 認証された接続のみが許可されます。

```
# inetadm -m svc:/network/ftp:default exec="/usr/sbin/in.ftpd -a"
```

3 **他のサービスを無効にします。**

in.rshd と in.rlogind デーモンを無効にする必要があります。

```
# svcadm disable network/shell
# svcadm disable network/login:rlogin
```

▼ KDC サーバーへのアクセスを制限する方法

マスター KDC およびスレーブ KDC には、KDC データベースのローカルコピーがあります。データベースを保護するためにこれらのサーバーへのアクセス権を制限することは、Kerberos 全体のセキュリティーにとって重要です。

1 **必要に応じて、遠隔サービスを無効にします。**

KDC サーバーをセキュリティー保護するために、不要なネットワークサービスをすべて無効にします。構成によっては、いくつかのサービスは既に無効になっています。svcs コマンドを使用して、サービス状態を確認します。ほとんどの環境では、KDC がマスターの場合に動作している必要のあるサービスは krb5kdc と kadmin のみです。ループバック TLI を使用するサービス (ticlts、 ticotsord、 および ticots) は、有効にしておくことができます。

```
# svcadm disable network/comsat
# svcadm disable network/dtspc/tcp
# svcadm disable network/finger
# svcadm disable network/login:rlogin
# svcadm disable network/rexec
# svcadm disable network/shell
# svcadm disable network/talk
# svcadm disable network/tname
# svcadm disable network/uucp
# svcadm disable network/rpc_100068_2-5/rpc_udp
```

2 **KDC をサポートするハードウェアに対するアクセスを制限します。**

物理的なアクセスを制限するために、KDC とそのモニターは安全な場所に設置します。このサーバーへのアクセスを完全に制限することが目的です。

3 **KDC データベースのバックアップを、ローカルディスクまたはスレーブ KDC に格納します。**

KDC のバックアップをテープに作成する場合、そのテープのセキュリティーを十分に確保してください。キータブファイルのコピーも、同様に作成します。これらのファイルをローカルファイルシステムに格納する場合は、できるだけほかのシステムと共有しないでください。格納先のファイルシステムは、マスター KDC または任意のスレーブ KDC から選択できます。

▼ 辞書ファイルを使用してパスワードセキュリティーを強化する方法

Kerberos サービスで辞書ファイルを使用することにより、新しい資格の作成時に辞書内の単語がパスワードとして使用されるのを防ぐことができます。辞書の用語がパスワードとして使用されないようにすると、パスワードの推測が困難になります。デフォルトでは `/var/krb5/kadm5.dict` ファイルが使用されますが、これは空です。

1 **マスター KDC 上でスーパーユーザーになります。**

2 **KDC 構成ファイル (`kdc.conf`) を編集します。**

行を追加して、サービスで辞書ファイルが使用されるようにする必要があります。この例では、`spell` ユーティリティーに含まれる辞書を使用します。構成ファイルの詳細は、`kdc.conf(4)` のマニュアルページを参照してください。

```
kdc1 # cat /etc/krb5/kdc.conf
[kdcdefaults]
        kdc_ports = 88,750

[realms]
        EXAMPLE.COM = {
                profile = /etc/krb5/krb5.conf
                database_name = /var/krb5/principal
                admin_keytab = /etc/krb5/kadm5.keytab
                acl_file = /etc/krb5/kadm5.acl
                kadmind_port = 749
                max_life = 8h 0m 0s
                max_renewable_life = 7d 0h 0m 0s
                sunw_dbprop_enable = true
                sunw_dbprop_master_ulogsize = 1000
                dict_file = /usr/share/lib/dict/words
                }
```

3 **Kerberos デーモンを再起動します。**

```
kdc1 # svcadm restart -r network/security/krb5kdc
kdc1 # svcadm restart -r network/security/kadmin
```

◆ ◆ ◆ 第 24 章

Kerberos エラーメッセージと障害追跡

この章では、Kerberos サービスを使用するときに発生するエラーメッセージの解決策を説明します。また、さまざまな問題を解決するためのヒントについても説明します。次に、この章で説明するエラーメッセージと障害追跡方法の一覧を示します。

Kerberos のエラーメッセージ

この節では、Kerberos のエラーメッセージ、エラーの発生原因、およびその対処方法について説明します。

SEAM ツールのエラーメッセージ

主体またはポリシーのリストにアクセスできません; 「名前」フィールドを使用してください。(Unable to view the list of principals or policies; use the Name field.)

原因: ログインに使用した admin 主体には、Kerberos ACL ファイル (kadm5.acl) のリスト特権 (l) がありません。このため、主体リストおよびポリシーリストを表示できません。

対処方法: 主体名およびポリシー名を「名前 (Name)」フィールドに入力するか、適切な特権を持つ主体を使用してログインする必要があります。

`JNI: Java array creation failed`

`JNI: Java class lookup failed`

`JNI: Java field lookup failed`

`JNI: Java method lookup failed`

`JNI: Java object lookup failed`

`JNI: Java object field lookup failed`

`JNI: Java string access failed`

`JNI: Java string creation failed`

原因: SEAM ツール (`gkadmin`) で使用される Java Native Interface で重大な問題が発生しました。

対処方法: `gkadmin` を終了して再起動してください。それでも問題が解決しない場合は、バグを報告してください。

Kerberos 共通エラーメッセージ (A - M)

この節では、Kerberos コマンド、Kerberos デーモン、PAM フレームワーク、GSS インタフェース、NFS サービス、および Kerberos ライブラリに共通するエラーメッセージを、英語版メッセージのアルファベット順 (A - M) に示します。

`All authentication systems disabled; connection refused` (すべての認証システムが無効です。接続が拒否されました。)

原因: このバージョンの `rlogind` は認証メカニズムをサポートしていません。

対処方法: `rlogind` の起動時に `-k` オプションが指定されていることを確認してください。

`Another authentication mechanism must be used to access this host` (このホストにアクセスするには、別の認証メカニズムを使用する必要があります。)

原因: 認証を実行できませんでした。

対処方法: クライアントが Kerberos V5 メカニズムを使って認証を行なっていることを確認してください。

`Authentication negotiation has failed, which is required for encryption. Good bye.` (暗号化に必要となる認証のネゴシエーションが失敗しました。処理を中断します。)

原因: 認証で、サーバーとのネゴシエーションが失敗しました。

対処方法: telnet と toggle authdebug コマンドを実行して認証デバッグ機能を開始し、そのデバッグメッセージからさらなる手掛かりを得てください。また、所有している資格が有効であることも確認してください。

Bad krb5 admin server hostname while initializing kadmin interface (kadmin インタフェースを初期化中に、krb5 admin サーバーホスト名が無効です。)
原因: krb5.conf ファイルの admin_server に、無効なホスト名が設定されています。

対処方法: krb5.conf ファイルの admin_server 行に、マスター KDC の正しいホスト名を指定してください。

Bad lifetime value (有効期限値が無効です。)
原因: 指定した有効期限値が無効または間違った形式です。

対処方法: 指定した値が、kinit(1) のマニュアルページの時刻形式のセクションに適合していることを確認してください。

Bad start time value (開始時刻の値が無効です。)
原因: 指定した開始時刻が無効または間違った形式です。

対処方法: 指定した値が、kinit(1) のマニュアルページの時刻形式のセクションに適合していることを確認してください。

Cannot contact any KDC for requested realm (要求されたレルムの KDC に接続できません。)
原因: 要求されたレルムの KDC が応答しません。

対処方法: 1 つ以上の KDC (マスターまたはスレーブ) にアクセスできること、または krb5kdc デーモンが KDC 上で動作していることを確認してください。/etc/krb5/krb5.conf ファイルに指定されている構成済みの KDC (kdc = *kdc_name*) を確認してください。

Cannot determine realm for host (ホスト用のレルムを決定できません。)
原因: Kerberos がホストのレルム名を判断できません。

対処方法: デフォルトのレルム名を指定するか、Kerberos 構成ファイル (krb5.conf) にドメイン名のマッピングを設定してください。

Cannot find KDC for requested realm (要求されたレルムの KDC が見つかりません。)
原因: 要求されたレルムに KDC が見つかりません。

対処方法: Kerberos 構成ファイル (krb5.conf) の realm セクションに KDC が指定されていることを確認してください。

`cannot initialize realm` *realm-name* (レルム `realm-name` を初期化できません。)
原因: KDC に stash ファイルが存在しない可能性があります。

対処方法: KDC に stash ファイルが存在することを確認してください。存在しない場合は、`kdb5_util` コマンドを使用して stash ファイルを作成し、再度 `krb5kdc` コマンドを実行します。

`Cannot resolve KDC for requested realm` (要求されたレルムの `KDC` を解決できません。)
原因: Kerberos がレルムの KDC を判断できません。

対処方法: Kerberos 構成ファイル (`krb5.conf`) の `realm` セクションに KDC が指定されていることを確認してください。

`Cannot reuse password` (パスワードは再利用できません。)
原因: 指定したパスワードは、以前にこの主体によって使用されています。

対処方法: 以前に使用されたことのないパスワードを選択してください。少なくとも KDC データベースに主体ごとに保持されている数のパスワードは、選択しないでください。このポリシーは、主体のポリシーによって適用されます。

`Can't get forwarded credentials` (転送された資格を取得できません。)
原因: 資格の転送ができません。

対処方法: この主体に転送可能な資格を設定してください。

`Can't open/find Kerberos configuration file` (Kerberos 構成ファイルを開けません / 見つかりません。)
原因: Kerberos 構成ファイル (`krb5.conf`) を使用できません。

対処方法: `krb5.conf` ファイルが、正しい場所に配置されていることを確認してください。また、このファイルに正しいアクセス権が与えられていることを確認してください。このファイルに対する書き込み権は `root`、 読み込み権はすべてのユーザーに与える必要があります。

`Client did not supply required checksum--connection rejected` (必要なチェックサム情報がクライアントから提供されませんでした。接続が拒否されました。)
原因: チェックサム付き認証で、クライアントとのネゴシエーションに失敗しました。クライアントが、初期接続をサポートしない旧 Kerberos V5 プロトコルを使用している可能性があります。

対処方法: クライアントが、初期接続をサポートする Kerberos V5 プロトコルを使用していることを確認してください。

`Client/server realm mismatch in initial ticket request` (初期チケット要求でクライアント/サーバーレルムが一致していません。)
原因: 初期チケット要求で、クライアントとサーバーのレルムが一致していません。

対処方法: 通信しているサーバーがクライアントと同じレルムに配置されていること、またはレルム構成が正しいことを確認してください。

`Client or server has a null key` (クライアントまたはサーバーの鍵が空です。)
原因: クライアントまたはサーバーの鍵が空です。

対処方法: `kadmin` の `cpw` コマンドを使用して、主体の鍵の値を入力してください。

`Communication failure with server while initializing kadmin interface` (kadmin インタフェースを初期化中に、サーバーとの通信の失敗です。)
原因: 管理サーバーとして指定したホスト (マスター KDC) 上で、`kadmind` デーモンが動作していません。

対処方法: マスター KDC に正しいホスト名が指定されていることを確認してください。ホスト名が正しい場合は、指定したマスター KDC 上で `kadmind` が動作していることを確認してください。

`Credentials cache file permissions incorrect` (資格キャッシュファイルのアクセス権が正しくありません。)
原因: 資格キャッシュ (`/tmp/krb5cc_`*uid*) に対する読み取り権または書き込み権が適切ではありません。

対処方法: 資格キャッシュに対する読み取り権および書き込み権があることを確認してください。

`Credentials cache I/O operation failed` *XXX* (資格キャッシュ入出力操作が失敗しました。`XXX`)
原因: システムの資格キャッシュ (`/tmp/krb5cc_`*uid*) に書き込むときに、Kerberos で問題が発生しました。

対処方法: 資格キャッシュが削除されていないことを確認し、`df` コマンドを使用してデバイスの空き領域を確認してください。

`Decrypt integrity check failed` (復号化で整合性チェックが失敗しました。)
原因: チケットが無効である可能性があります。

対処方法: 次の条件をいずれも確認してください。

- 資格が有効であることを確認してください。`kdestroy` を使用してチケットを破棄し、`kinit` を使用して新しいチケットを作成します。

- 対象ホストのキータブファイルに対して、正しいバージョンのサービス鍵が割り当てられていることを確認してください。kadmin を使用して、Kerberos データベースのサービス主体 (host/*FQDN-hostname* など) の鍵バージョン番号を表示します。対象ホスト上で klist -k を使用して、鍵バージョン番号がその番号であることを確認します。

`Encryption could not be enabled. Good bye.` (暗号化を有効化できませんでした。処理を中止します。)

原因: 暗号化で、サーバーとのネゴシエーションに失敗しました。

対処方法: telnet と toggle encdebug コマンドを実行して認証デバッグ機能を開始し、そのデバッグメッセージからさらなる手掛かりを得てください。

`failed to obtain credentials cache` (資格キャッシュを取得できませんでした。)

原因: kadmin の初期化中に、kadmin が admin 主体の資格を取得しようとしましたが、失敗しました。

対処方法: kadmin を実行したときに、正しい主体とパスワードを使用したことを確認してください。

`Field is too long for this implementation` (この実装ではフィールドが長すぎます。)

原因: Kerberos アプリケーションから送信されたメッセージのサイズが長すぎます。このエラーは、トランスポートプロトコルが UDP の場合に発生します。UDP では、デフォルトの最大メッセージ長は 65535 バイトです。また、Kerberos サービスから送信されるプロトコルメッセージの各フィールドにも制限があります。

対処方法: KDC サーバーの /etc/krb5/kdc.conf ファイルでトランスポートを UDP に制限していないことを確認してください。

`GSS-API (or Kerberos) error` (GSS-API (または Kerberos) エラー)

原因: このメッセージは、汎用 GSS-API または Kerberos のエラーメッセージで、いくつかの問題の組み合わせによって発生した可能性があります。

対処方法: /var/krb5/kdc.log ファイルを確認して、このエラーが発生したときに詳細なエラーメッセージが記録されているかどうかを確認してください。

`Hostname cannot be canonicalized` (ホスト名を展開できません。)

原因: Kerberos クライアントはサーバーの完全修飾ホスト名を見つけることができません。

対処方法: このサーバーホスト名が DNS に定義されていることと、ホスト名とアドレス間の双方向のマッピングについて整合性を確認してください。

`Illegal cross-realm ticket` (レルム間のチケットが無効です。)
原因: 送信されたチケットのレルム間関係が正しくありません。レルム間に正しい信頼関係が設定されていない可能性があります。

対処方法: 使用しているレルム間の信頼関係が正しいことを確認してください。

`Improper format of Kerberos configuration file` (Kerberos 構成ファイルのフォーマットが不適切です。)
原因: Kerberos 構成ファイルに無効なエントリがあります。

対処方法: `krb5.conf` ファイル内のすべての関係式に、= 記号と値が使用されていることを確認してください。また、各下位セクションが角カッコで囲まれていることも確認してください。

`Inappropriate type of checksum in message` (メッセージのチェックサムのタイプが不適切です。)
原因: このメッセージに無効なチェックサムタイプが含まれています。

対処方法: `krb5.conf` および `kdc.conf` ファイルに指定されているチェックサムタイプが有効であることを確認してください。

`Incorrect net address` (ネットアドレスが間違っています。)
原因: ネットワークアドレスが一致しません。転送されたチケット内のネットワークアドレスが、チケットが処理されたときのネットワークアドレスと一致しません。このメッセージは、チケットの転送時に発生します。

対処方法: ネットワークアドレスが正しいことを確認してください。`kdestroy` を使用してチケットを破棄し、`kinit` を使用して新しいチケットを作成します。

`Invalid credential was supplied` (無効な資格が指定されました。)

`Service key not available` (サービス鍵が使用できません。)
原因: 資格キャッシュ内のサービスチケットが間違っている可能性があります。

対処方法: 現在の資格キャッシュを破棄して、このサービスを使用する前に `kinit` を再実行してください。

`Invalid flag for file lock mode` (ファイルロックモードのフラグが無効です。)
原因: Kerberos の内部エラーが発生しました。

対処方法: バグを報告してください。

`Invalid message type specified for encoding` (符号化に対し無効なメッセージタイプが指定されました。)
原因: Kerberos アプリケーションから送信されたメッセージ形式を、Kerberos が認識できません。

対処方法: 使用するサイトまたはベンダーで開発したKerberosアプリケーションを使用している場合は、Kerberosが正しく使用されていることを確認してください。

`Invalid number of character classes` (文字クラス数が正しくありません。)
原因: 主体に指定したパスワードに、主体のポリシーによって適用された数のパスワードクラスが含まれていません。

対処方法: ポリシーに指定されている最小パスワードクラス数を使用して、パスワードを指定してください。

`KADM err: Memory allocation failure` (KADM エラー: メモリー割り当ての失敗です。)
原因: `kadmin` の実行に必要なメモリーが不足しています。

対処方法: メモリーを解放してから、`kadmin` を再実行してください。

`kadmin: Bad encryption type while changing host/<FQDN>'s key` (host/<FQDN> のキーの変更中に不正な暗号化タイプが見つかりました。)
原因: Solaris 10 8/07 リリースの基本リリースには、デフォルトの暗号化タイプがいくつか追加されました。以前のバージョンのソフトウェアを実行しているKDCがサポートしない暗号化タイプをクライアントが要求する場合があります。

対処方法: この問題を解決するための対処方法がいくつか用意されています。もっとも簡単に実行できる方法を次に示します。

1. SUNWcryおよびSUNWcryrパッケージをKDCサーバーに追加します。これにより、KDCがサポートする暗号化タイプの数が増えます。
2. `aes256` 暗号化タイプを含まないように、クライアント上の `krb5.conf` の `permitted_enctypes` を設定します。この手順は、新しいクライアントが追加されるごとに実行する必要があります。

`KDC can't fulfill requested option` (KDC は要求したオプションを処理できません。)
原因: 要求されたオプションをKDCが許可しませんでした。遅延または転送可能オプションが要求されましたが、KDCが許可しませんでした。または、TGTの更新が要求されましたが、更新可能なTGTが存在しない可能性があります。

対処方法: KDCが許可しないオプションまたは使用できない種類のチケットを要求していないかどうかを確認してください。

`KDC policy rejects request` (KDC ポリシーは要求を拒否します。)
原因: KDCポリシーが要求を許可しませんでした。たとえば、KDCに対する要求にIPアドレスが含まれていないことがあります。あるいは、要求は転送されたが、KDCが許可しなかった可能性があります。

対処方法: 正しいオプションを指定して kinit を実行していることを確認してください。必要に応じて、主体に関連付けられたポリシーを変更するか、要求が許可されるように主体の属性を変更します。ポリシーまたは主体を変更するには、kadmin を使用します。

KDC 返信が期待したものと一致しません
原因: KDC の応答に予期した主体名が含まれていないか、応答内のその他の値が正しくありません。

対処方法: 通信先の KDC が RFC4120 に準拠していること、送信している要求が Kerberos V5 要求であること、または KDC が有効であることを確認してください。

kdestroy:Could not obtain principal name from cache (キャッシュから主体名を取得できません。)
原因: 資格キャッシュが欠落しているか、または破壊されています。

対処方法: 指定したキャッシュ位置が正しいことを確認してください。必要に応じて、kinit を使用して削除し、新しい TGT を取得してください。

kdestroy:Could not obtain principal name from cache (キャッシュの破棄中に資格キャッシュが見つかりませんでした。)
原因: 資格キャッシュ (/tmp/krb5c_*uid*) が欠落しているか、または破壊されています。

対処方法: 指定したキャッシュ位置が正しいことを確認してください。必要に応じて、kinit を使用して削除し、新しい TGT を取得してください。

kdestroy:Could not obtain principal name from cache (TGT 期限切れの警告が削除されません。)
原因: 資格キャッシュが欠落しているか、または破壊されています。

対処方法: 指定したキャッシュ位置が正しいことを確認してください。必要に応じて、kinit を使用して削除し、新しい TGT を取得してください。

Kerberos authentication failed (Kerberos 認証に失敗しました。)
原因: Kerberos パスワードが正しくないか、または UNIX パスワードと一致していません。

対処方法: パスワードが一致していない場合は、Kerberos 認証に成功する別のパスワードを指定する必要があります。ユーザーが元のパスワードを忘れた可能性があります。

Kerberos V5 refuses authentication (Kerberos V5 によって認証が拒否されました。)
原因: 認証で、サーバーとのネゴシエーションが失敗しました。

対処方法: telnet と toggle authdebug コマンドを実行して認証デバッグ機能を開始し、そのデバッグメッセージからさらなる手掛かりを得てください。また、所有している資格が有効であることも確認してください。

Key table entry not found (鍵テーブルエントリが見つかりません。)
原因: ネットワークアプリケーションサーバーのキータブファイルに、サービス主体のエントリがありません。

対処方法: サーバーのキータブファイルに適切なサービス主体を追加して、Kerberos サービスを提供できるようにしてください。

Key version number for principal in key table is incorrect (鍵テーブルの主体の鍵バージョン番号が正しくありません。)
原因: キータブファイルと Kerberos データベース内の主体の鍵バージョンが異なります。サービスの鍵が変更されたか、旧サービスチケットを使用している可能性があります。

対処方法: kadmin などによってサービスの鍵が変更されている場合は、新しい鍵を抽出して、サービスが動作しているホストのキータブファイルに格納する必要があります。

または、旧サービスチケットを使用しているため、鍵が古い可能性があります。kdestroy コマンドを実行し、次に kinit コマンドを再度実行してください。

kinit: gethostname failed (gethostname が失敗しました。)
原因: ローカルネットワーク構成でのエラーは、kinit が失敗する原因になります。

対処方法: ホストが正しく構成されていることを確認してください。

login: load_modules: can not open module /usr/lib/security/pam_krb5.so.1
(load_modules: /usr/lib/security/pam_krb5.so.1 モジュールを開けません。)
原因: Kerberos PAM モジュールが存在しないか、有効な実行可能バイナリではありません。

対処方法: Kerberos PAM モジュールが /usr/lib/security ディレクトリに存在し、有効な実行可能バイナリであることを確認してください。また、/etc/pam.conf ファイルに pam_krb5.so.1 への正しいパスが指定されていることも確認してください。

Looping detected inside krb5_get_in_tkt (krb5_get_in_tkt 内部でループが検出されました。)
原因: Kerberos が初期チケットを複数回取得しようとしましたが、失敗しました。

対処方法: 認証要求に対して 1 つ以上の KDC が応答していることを確認してください。

`Master key does not match database` (マスター鍵がデータベースと一致しません。)
原因: 読み込まれたデータベースのダンプが、マスター鍵を含むデータベースから作成されませんでした。マスター鍵は `/var/krb5/.k5.`*REALM* 内に格納されています。

対処方法: 読み込まれたデータベースダンプ内のマスター鍵が、`/var/krb5/.k5.`*REALM* に配置されているマスター鍵と一致していることを確認してください。

`Matching credential not found` (一致する資格が見つかりません。)
原因: 要求に一致する資格が見つかりませんでした。資格キャッシュで使用できない資格を要求しています。

対処方法: `kdestroy` を使用してチケットを破棄し、`kinit` を使用して新しいチケットを作成します。

`Message out of order` (メッセージの順序が違います。)
原因: 順次送信されたメッセージが順不同で着信しました。一部のメッセージが転送中に失われました。

対処方法: Kerberos セッションを再度初期化してください。

`Message stream modified` (メッセージストリームが変更されました。)
原因: 計算されたチェックサムとメッセージのチェックサムが一致しませんでした。転送中のメッセージが変更された可能性があります。セキュリティー侵害が発生している可能性があります。

対処方法: メッセージがネットワーク経由で正しく送信されていることを確認してください。このメッセージが送信中に改変された可能性もあるため、`kdestroy` を使用してチケットを破棄し、使用している Kerberos サービスを再度初期化してください。

Kerberos 共通エラーメッセージ (N - Z)

この節では、Kerberos コマンド、Kerberos デーモン、PAM フレームワーク、GSS インタフェース、NFS サービス、および Kerberos ライブラリに共通するエラーメッセージを、英語版メッセージのアルファベット順 (N - Z) に示します。

`No credentials cache file found` (資格キャッシュファイルが見つかりません。)
原因: Kerberos が資格キャッシュ (`/tmp/krb5cc_`*uid*) を見つけることができません。

対処方法: 資格ファイルが存在し、読み込み可能であることを確認してください。存在しない場合は、`kinit` を再度実行します。

`No credentials were supplied, or the credentials were unavailable or inaccessible` (資格が提供されていません。あるいは、資格を使用またはアクセスできません。)

`No credentials cache found` (資格キャッシュが見つかりません。)

原因: ユーザーの資格キャッシュが間違っているか、または存在しません。

対処方法: ユーザーは、サービスを開始する前に、`kinit` を実行する必要があります。

`No credentials were supplied, or the credentials were unavailable or inaccessible` (資格が提供されていません。あるいは、資格を使用またはアクセスできません。)

`No principal in keytab matches desired name` (キータブ内の主体が目的の名前と一致しません。)

原因: サーバーの認証中にエラーが発生しました。

対処方法: ホスト主体またはサービス主体が、サーバーのキータブファイル内にあることを確認してください。

`Operation requires "`*privilege*`" privilege` (操作には `privilege` 特権が必要です。)

原因: 使用された `admin` 主体に対して、`kadm5.acl` ファイルに設定されている適切な特権が割り当てられていません。

対処方法: 適切な特権を持つ主体を使用してください。または、`kadm5.acl` ファイルを変更して、使用した主体に適切な特権を割り当てます。通常は、名前の一部に `/admin` が含まれる主体には、適切な特権が割り当てられています。

`PAM-KRB5 (auth): krb5_verify_init_creds failed: Key table entry not found` (`PAM-KRB5 (auth): krb5_verify_init_creds` に失敗しました: `Key table entry not found` (鍵テーブルエントリが見つかりません。)

原因: 遠隔アプリケーションは、ローカル `/etc/krb5/krb5.keytab` ファイル内にあるホストのサービス主体を読み込もうとしましたが、サービス主体が存在しませんでした。

対処方法: ホストのキータブファイルにホストのサービス主体を追加してください。

`Password is in the password dictionary` (パスワードはパスワード辞書にあります。)

原因: 指定したパスワードが使用中のパスワード辞書にすでに存在します。選択したパスワードが適切ではありません。

対処方法: パスワードクラスを組み合わせたパスワードを選択してください。

`Permission denied in replay cache code` (再実行キャッシュコードでアクセス権がありません。)
原因: システムの再実行キャッシュを開けませんでした。サーバーは、現在のユーザー ID と異なるユーザー ID で最初に実行された可能性があります。

対処方法: 再実行キャッシュに適切なアクセス権が割り当てられていることを確認してください。再実行キャッシュは、Kerberos サーバーアプリケーションが動作するホストに格納されます。再実行キャッシュファイルは、`root` ではないユーザーの場合は `/var/krb5/rcache/rc_`*service_name_uid* という名前です。root ユーザーの再実行キャッシュは、 `/var/krb5/rcache/root/rc_`*service_name* です。

`Protocol version mismatch` (プロトコルバージョンが一致していません。)
原因: Kerberos V4 要求が KDC に送信された可能性があります。Kerberos サービスでは、Kerberos V5 プロトコルだけがサポートされます。

対処方法: アプリケーションが Kerberos V5 プロトコルを使用していることを確認してください。

`Request is a replay` (要求は再送です。)
原因: この要求は、すでにこのサーバーに送信され、処理が完了しています。チケットが盗まれた可能性があり、ほかのユーザーがチケットを再使用しようとしています。

対処方法: しばらくしてから要求を再発行してください。

`Requested principal and ticket don't match` (要求した主体とチケットは一致しません。)
原因: 接続するサービス主体と使用するサービスチケットが一致しません。

対処方法: DNS が適切に機能することを確認してください。別のベンダーのソフトウェアを使用する場合は、そのソフトウェアが主体名を正しく使用していることを確認します。

`Requested protocol version not supported` (要求したプロトコルバージョンはサポートされていません。)
原因: Kerberos V4 要求が KDC に送信された可能性があります。Kerberos サービスでは、Kerberos V5 プロトコルだけがサポートされます。

対処方法: アプリケーションが Kerberos V5 プロトコルを使用していることを確認してください。

`Server refused to negotiate authentication, which is required for encryption. Good bye.`
原因: 遠隔アプリケーションは、クライアントからの Kerberos 認証を受け取ることができないか、またはそのように構成されていません。

対処方法: 認証をネゴシエーションできるアプリケーションを提供するか、認証を有効にする適切なフラグを使用してアプリケーションを設定します。

`Server refused to negotiate encryption. Good bye.`
原因: 暗号化で、サーバーとのネゴシエーションに失敗しました。

対処方法: `telnet` と `toggle encdebug` コマンドを実行して認証デバッグ機能を開始し、そのデバッグメッセージからさらなる手掛かりを得てください。

`Server rejected authentication (during sendauth exchange)` (サーバーが認証を拒否しました (`sendauth` 交換で)。)
原因: 通信しようとしているサーバーが認証を拒否しました。ほとんどの場合、このエラーは Kerberos データベースを伝播するときに発生します。`kpropd.acl` ファイル、DNS、 またはキータブファイルに問題が発生している可能性があります。

対処方法: `kprop` 以外のアプリケーションを実行しているときにこのエラーが発生した場合は、サーバーのキータブファルが正しいかどうかを調査してください。

`The ticket isn't for us` (チケットはわれわれのものではありません。)

`Ticket/authenticator don't match` (チケット/ オーセンティケータが一致しません。)
原因: チケットとオーセンティケータが一致しません。要求内の主体名がサービス主体の名前と一致しなかった可能性があります。その原因は、サービスが非 FQDN 名を期待しているときにチケットが主体の FQDN 名とともに送信された、サービスが FQDN 名を期待しているときに非 FDQN 名が送信された、のいずれかです。

対処方法: `kprop` 以外のアプリケーションを実行しているときにこのエラーが発生した場合は、サーバーのキータブファルが正しいかどうかを調査してください。

`Ticket expired` (チケットの有効期限が切れました。)
原因: チケットが期限切れになっています。

対処方法: `kdestroy` を使用してチケットを破棄し、`kinit` を使用して新しいチケットを作成します。

`Ticket is ineligible for postdating` (チケットには遅延処理の資格がありません。)
原因: この主体は、チケットの遅延を許可していません。

対処方法: `kadmin` を使用して主体を変更し、遅延を許可してください。

`Ticket not yet valid` (チケットはまだ有効ではありません。)
原因: 遅延チケットはまだ有効ではありません。

対処方法: 正しい日付で新しいチケットを作成するか、現在のチケットが有効になるまで待ちます。

`Truncated input file detected` (不完全な入力ファイルを検出しました。)
原因: 操作に使用されたデータベースダンプファイルが完全ではありません。

対処方法: ダンプファイルを作成し直すか、別のデータベースダンプファイルを使用します。

`Unable to securely authenticate user ... exit` (ユーザーを安全に認証できません。処理を終了します。)
原因: 認証で、サーバーとのネゴシエーションが失敗しました。

対処方法: `telnet` と `toggle authdebug` コマンドを実行して認証デバッグ機能を開始し、そのデバッグメッセージからさらなる手掛かりを得てください。また、所有している資格が有効であることも確認してください。

`Wrong principal in request` (要求した主体は正しくありません。)
原因: チケットの主体名が無効です。DNS または FQDN の問題が発生している可能性があります。

対処方法: サービスの主体とチケットの主体が一致していることを確認してください。

Kerberos の障害追跡

この節では、Kerberos ソフトウェアの障害追跡について説明します。

krb5.conf ファイルの書式の問題

`krb5.conf` ファイルの書式が正しくない場合、次のエラーメッセージが端末またはログファイルに表示されることがあります。

```
Improper format of Kerberos configuration file while initializing krb5 library
```

`krb5.conf` ファイルの書式に問題があると、関連するサービスが攻撃を受けやすくなる可能性があります。この問題を解決しないと、Kerberos 機能を使用できません。

Kerberos データベースの伝播の問題

Kerberos データベースの伝播が失敗した場合、スレーブ KDC とマスター KDC との間で、およびマスター KDC からスレーブ KDC サーバーへ、`/usr/bin/rlogin -x` を試してみてください。

KDC がアクセスを制限するように設定されていた場合、 rlogin が無効となり、このコマンドを使って問題を解決することができません。KDC で rlogin を有効にするには、eklogin サービスを有効にする必要があります。

```
# svcadm enable svc:/network/login:eklogin
```

問題の障害追跡を終了したあと、eklogin サービスを無効にする必要があります。

rlogin が正常に動作しなかった場合、問題の原因は KDC のキータブファイルにある可能性が高くなります。rlogin が正常に動作した場合、問題の原因はキータブファイルやネームサービスにはありません。というのも、rlogin と伝播ソフトウェアは同じ host/*host-name* 主体を使用しているからです。この場合、kpropd.acl ファイルが正しいことを確認してください。

Kerberos NFS ファイルシステムのマウントの問題

- Kerberos NFS ファイルシステムのマウントに失敗した場合には、NFS サーバーに /var/rcache/root ファイルが存在することを確認してください。ファイルシステムの所有者が root でない場合は、削除してから再度マウントします。
- Kerberos NFS ファイルシステムへのアクセスに問題がある場合は、使用するシステムと NFS サーバー上で gssd サービスが有効になっていることを確認してください。
- Kerberos NFS ファイルシステムにアクセスしようとしたときに invalid argument または bad directory のエラーメッセージが表示された場合は、NFS ファイルシステムをマウントするときに完全指定形式の DNS 名を使用していない可能性があります。マウントされているホストが、サーバーのキータブファイル内のサービス主体名に含まれるホスト名と一致していません。

  また、複数の Ethernet インタフェースを実装したサーバーに DNS を設定するときに、ホスト単位に複数のアドレスレコードを割り当てずに、インタフェース単位に名前を割り当てた場合にも、この問題が発生します。Kerberos サービスの場合は、次のようにホスト単位に複数のアドレスレコードを設定する必要があります。[1]:

```
my.host.name.   A       1.2.3.4
                A       1.2.4.4
                A       1.2.5.4

my-en0.host.name.       A       1.2.3.4
my-en1.host.name.       A       1.2.4.4
my-en2.host.name.       A       1.2.5.4

4.3.2.1         PTR     my.host.name.
4.4.2.1         PTR     my.host.name.
4.5.2.1         PTR     my.host.name.
```

[1] Ken Hornstein, “Kerberos FAQ,” [http://www.cmf.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html#kerbdns], accessed 10 March 2010.

この例の設定では、インタフェースごとに 1 つの参照が割り当てられます。また、サーバーのキータブファイル内で、3 つのサービス主体の代わりに、1 つのサービス主体を使用できます。

root の認証の問題

使用するシステムのスーパーユーザーになるときの認証に失敗し、ホストのキータブファイルに root 主体がすでに追加されている場合は、2 つの問題を確認する必要があります。まず、キータブファイル内の root 主体が、そのインスタンスとして完全指定形式名であることを確認します。完全指定形式名の場合は、/etc/resolv.conf ファイルを確認して、システムが DNS クライアントとして正しく設定されていることを確認してください。

GSS 資格の UNIX 資格へのマッピングの監視

資格マッピングを監視するには、最初に、/etc/gss/gsscred.conf ファイルの次の行をコメント解除します。

```
SYSLOG_UID_MAPPING=yes
```

次に、gssd サービスに/etc/gss/gsscred.conf ファイルから情報を取得するように指示します。

```
# pkill -HUP gssd
```

これで、gssd が資格マッピングを要求するときにそれを監視できるようになります。syslog.conf ファイルが debug 重要度を扱う auth システム機能用に構成されている場合、syslogdを使用してマッピングを記録できます。

◆ ◆ ◆ 第 25 章

Kerberos 主体とポリシーの管理 (手順)

この章では、主体とそれに関連するポリシーを管理する手順について説明します。また、ホストのキータブファイルの管理方法についても説明します。

この章は、主体とポリシーを管理する必要のあるユーザーを対象にしています。主体とポリシーを計画するときの考慮事項など、主体とポリシーについて理解している必要があります。第 21 章「Kerberos サービスについて」および第 22 章「Kerberos サービスの計画」をそれぞれ参照してください。

この章で説明する情報は次のとおりです。

Kerberos 主体とポリシーの管理方法

マスター KDC の Kerberos データベースには、使用するレルムの Kerberos 主体、そのパスワード、ポリシーなどの管理情報がすべて含まれています。主体を作成または削除したり、主体の属性を変更したりするには、`kadmin` または `gkadmin` コマンドのいずれかを使用します。

`kadmin` コマンドには、対話型のコマンド行インタフェースが用意されています。このインタフェースを使用して、Kerberos 主体、ポリシー、およびキータブファイルを管理することができます。`kadmin` コマンドには、次の 2 つの種類があります。

- `kadmin` – Kerberos 認証を使用して、ネットワーク上の任意の場所から安全に操作できます
- `kadmin.local` – マスター KDC 上で直接実行する必要があります

Kerberos を使用してユーザーを認証する点を除いて、2 つの kadmin の機能は同じです。kadmin に必要なデータベースを設定するときは、kadmin.local を使用します。

Oracle Solaris リリースには SEAM ツール (gkadmin) も用意されています。このツールは対話型のグラフィカルユーザーインタフェース (GUI) で、基本的に kadmin コマンドと同じ機能を持ちます。詳細は、522 ページの「SEAM Tool」を参照してください。

SEAM Tool

SEAM ツール (gkadmin) は、対話型グラフィカルユーザーインタフェース (GUI) で、Kerberos 主体とポリシーを管理することができます。このツールは、kadmin コマンドと同じ機能を持ちます。ただし、キータブファイルの管理はサポートしません。キータブファイルを管理するには、kadmin コマンドを使用する必要があります (551 ページの「キータブファイルの管理」を参照)。

kadmin コマンドと同様に、SEAM Toolは、Kerberos 認証と暗号化された RPC を使用して、ネットワーク上の任意の場所から安全に操作することができます。SEAM ツールでは、次の操作を行うことができます。

- デフォルト値または既存の主体に基づく新しい主体を作成する。
- 既存のポリシーに基づく新しいポリシーを作成する。
- 主体のコメントを追加する。
- 新しい主体を作成するときのデフォルト値を設定する。
- ツールを終了しないで別の主体としてログインする。
- 主体一覧とポリシー一覧を印刷または保存する。
- 主体一覧とポリシー一覧を表示および検索する。

SEAM ツールでは、コンテキストヘルプと一般的なオンラインヘルプも利用できます。

SEAM ツールを使用して実行できる操作について、次の作業マップで説明します。

- 526 ページの「Kerberos 主体の管理 (作業マップ)」
- 539 ページの「Kerberos ポリシーの管理 (作業マップ)」

また、SEAM ツールで指定または表示できる主体属性とポリシー属性については、547 ページの「SEAM ツールパネルの説明」を参照してください。

SEAM ツールに対応するコマンド行

この節では、SEAM ツールと同じ機能を提供する kadmin コマンドを示します。これらのコマンドは、X ウィンドウシステムで実行しなくても使用できます。この章のほとんどの手順では、SEAM ツールを使用します。ただし、多くの手順では、対応するコマンド行の使用例も挙げています。

表 25–1　SEAM ツールに対応するコマンド行

SEAM ツールの手順	対応する kadmin コマンド
主体の一覧を表示します。	list_principals または get_principals
主体の属性を表示します。	get_principal
新しい主体を作成します。	add_principal
主体を複製します。	対応するコマンド行なし
主体を変更します。	modify_principal または change_password
主体を削除します。	delete_principal
新しい主体を作成するときのデフォルトを設定します。	対応するコマンド行なし
ポリシーの一覧を表示します。	list_policies または get_policies
ポリシーの属性を表示します。	get_policy
新しいポリシーを作成します。	add_policy
ポリシーを複製します。	対応するコマンド行なし
ポリシーを変更します。	modify_policy
ポリシーを削除します。	delete_policy

SEAM ツールにより変更されるファイル

SEAM ツールが変更するファイルは、$HOME/.gkadmin ファイルだけです。このファイルには、新しい主体を作成するときのデフォルト値が含まれます。このファイルを更新するには、「Edit」メニューから「Properties」を選択します。

SEAM ツールの印刷機能とオンラインヘルプ機能

SEAM ツールには、印刷機能とオンラインヘルプ機能が用意されています。「Print」メニューから、次の要素をプリンタまたはファイルに送信できます。

- 指定したマスター KDC で使用できる主体の一覧
- 指定したマスター KDC で使用できるポリシーの一覧
- 現在選択されている主体または読み込まれている主体
- 現在選択されているポリシーまたは読み込まれているポリシー

「Help」メニューから、コンテキストヘルプと通常のヘルプを使用できます。「Help」メニューから「Context-Sensitive Help」を選択すると、「Context-Sensitive Help」ウィンドウが表示され、ツールがヘルプモードに切り

替わります。ヘルプモードのウィンドウで、任意のフィールド、ラベル、またはボタンをクリックすると、「Help」ウィンドウにその項目のヘルプが表示されます。ツールの通常モードに戻るには、「Help」ウィンドウで「Dismiss」をクリックします。

また、「Help Contents」を選択すると、HTML ブラウザが開き、この章で説明している概要や操作情報が表示されます。

SEAM ツールで大規模な一覧を使用する

登録した主体とポリシーが増加するにつれて、SEAM ツールが主体とポリシーを読み込んでそれらの一覧を表示する時間が長くなります。このため、ツールによる作業効率が低下します。この問題には、いくつかの対応方法があります。

まず、一覧を読み込む時間を完全になくすために、SEAM ツールに一覧を読み込まないようにします。この方法を設定するには、「Edit」メニューから「Properties」を選択し、「Show Lists」フィールドのチェックマークをはずします。一覧を読み込まない場合、一覧は表示されないため、一覧を使用して主体またはポリシーを選択できなくなります。代わりに、表示された新しい「Name」フィールドに主体またはポリシー名を入力し、その主体またはポリシーに適用する操作を選択する必要があります。名前を入力する操作は、一覧から項目を選択する操作と同じ効果を持ちます。

大規模な一覧を使用するときは、キャッシュを利用することもできます。SEAM ツールのデフォルトの動作として、一定量の一覧がキャッシュに格納されるように設定されています。SEAM ツールは、最初に一覧をキャッシュに読み込む必要がありますが、そのあとは一覧を再度読み込まずにキャッシュを使用できます。この方法では、サーバーから時間をかけて何回も一覧を読み込む必要がありません。

一覧がキャッシュに格納されるように設定するには、「Edit」メニューから「Properties」を選択します。キャッシュの設定には、次の 2 つの方法があります。一覧をキャッシュに永続的に格納するか、制限時間を指定します。制限時間を指定した場合は、その時間が経過すると、ツールはサーバーの一覧をキャッシュに再度読み込みます。

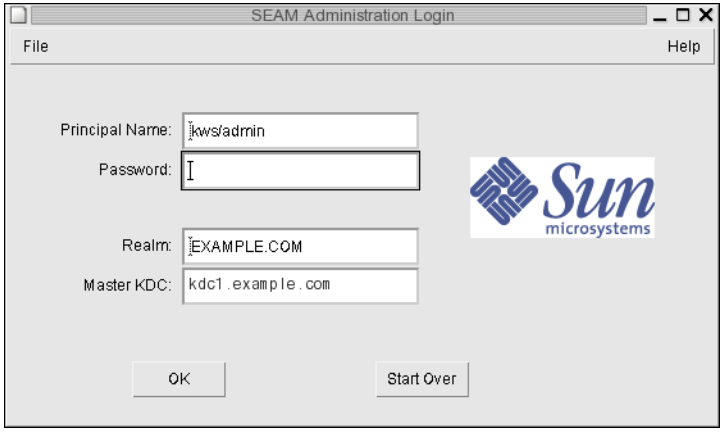
一覧をキャッシュに格納しても、一覧から主体とポリシーを選択することができます。このため、一覧を読み込まない最初の方法と異なり、SEAM ツールの利用には影響しません。また、キャッシュを利用した場合でも、ほかの主体とポリシーの変更を確認できなくなることがあります。ただし、使用している主体とポリシーを変更したときは最新の一覧が表示されます。主体とポリシーを変更すると、サーバーとキャッシュの一覧が更新されるためです。キャッシュを更新して、ほかの主体とポリシーの変更を確認し、最新の一覧を取得するには、任意のタイミングで「Refresh」メニューを使用します。サーバーから一覧が読み込まれ、キャッシュを更新することができます。

▼ SEAM ツールを起動する方法

1 **gkadmin** コマンドを使用して **SEAM** ツールを起動します。

```
$ /usr/sbin/gkadmin
```

「SEAM Login」ウィンドウが表示されます。



2 デフォルト値を使用しない場合は、新しいデフォルト値を指定します。

ウィンドウには、デフォルト値が自動的に表示されています。デフォルトの主体名は、`USER` 環境変数の現在の ID に `/admin` が付加されて作成されます (*username*`/admin`)。デフォルトの「Realm」フィールドおよび「Master KDC」フィールドは、`/etc/krb5/krb5.conf` ファイルから選択されます。デフォルト値を再度取得する場合は、「Start Over」をクリックします。

注 – 各「Principal Name」が実行できる管理操作は、Kerberos ACL ファイルの `/etc/krb5/kadm5.acl` で規定されます。権限の制限については、550 ページの「Kerberos 管理権限を制限して SEAM ツールを使用する」を参照してください。

---

3 指定した主体名のパスワードを入力します。

4 「了解 (OK)」をクリックします。

ウィンドウが表示され、すべての主体が示されます。

Kerberos 主体の管理

この節では、SEAM ツールを使用して主体を管理する手順について説明します。また、対応するコマンド行がある場合は、その例も示します。

Kerberos 主体の管理 (作業マップ)

作業	説明	参照先
主体の一覧を表示します。	「Principals」タブをクリックして、主体の一覧を表示します。	527 ページの「Kerberos 主体の一覧を表示する方法」
主体の属性を表示します。	「Principal List」の「Principal」を選択し、「Modify」ボタンをクリックして、主体の属性を表示します。	529 ページの「Kerberos 主体の属性を表示する方法」
新しい主体を作成します。	「Principal List」パネルの「Create New」ボタンをクリックして、新しい主体を作成します。	531 ページの「新しい Kerberos 主体を作成する方法」
主体を複製します。	「Principal List」から複製する主体を選択し、「Duplicate」ボタンをクリックして、主体を複製します。	534 ページの「Kerberos 主体を複製する方法」
主体を変更します。	「Principal List」から変更する主体を選択し、「Modify」ボタンをクリックして、主体を変更します。 主体名は変更できません。主体名を変更するときは、主体を複製し、新しい名前を指定して保存してから、古い主体を削除する必要があります。	534 ページの「Kerberos 主体を変更する方法」
主体を削除します。	「Principal List」から削除する主体を選択し、「Delete」ボタンをクリックして、主体を削除します。	535 ページの「Kerberos 主体を削除する方法」
新しい主体を作成するときのデフォルトを設定します。	「Edit」メニューから「Properties」を選択して、新しい主体を作成するときのデフォルトを設定します。	536 ページの「新しい Kerberos 主体を作成するときのデフォルトを設定する方法」
Kerberos 管理権限 (`kadm5.acl` ファイル) を変更します。	コマンド行のみ。Kerberos 管理権限により、主体が Kerberos データベースに対して実行できる操作 (追加、変更など) が決定されます。 各主体の Kerberos 管理権限を変更するときは、`/etc/krb5/kadm5.acl` ファイルを編集する必要があります。	537 ページの「Kerberos 管理権限を変更する方法」

新しい Kerberos 主体の自動作成

SEAM Toolは簡単に使用できますが、新しい主体を自動作成することができません。10 個または 100 個などの新しい主体を短時間で作成する場合は、自動作成を利用すると便利です。Bourne シェルスクリプトで kadmin.local コマンドを使用すると、主体を自動作成できます。

次のシェルスクリプト行は、新しい主体を自動作成する方法の例を示します。

```
awk '{ print "ank +needchange -pw", $2, $1 }' < /tmp/princnames |
        time /usr/sbin/kadmin.local> /dev/null
```

この例は、見やすいように 2 行に分割しています。このスクリプトは、princnames というファイルを読み込んで、そこに含まれている主体名とそのパスワードを Kerberos データベースに追加します。princnames ファイルをあらかじめ作成する必要があります。このファイルの各行には、主体とそのパスワードを 1 つ以上の空白で区切って指定します。主体に +needchange オプションを指定すると、ユーザーがその主体を使用して初めてログインしたときに、新しいパスワードを要求するプロンプトが表示されます。この方法を使用すると、princnames ファイル内のパスワードのセキュリティーが向上します。

より複雑なスクリプトも作成できます。たとえば、ネームサービスの情報を使用して、主体名に対応するユーザー名の一覧を取得できます。必要な作業とその方法は、使用環境要件とスクリプト使用技術によって決まります。

▼ Kerberos 主体の一覧を表示する方法

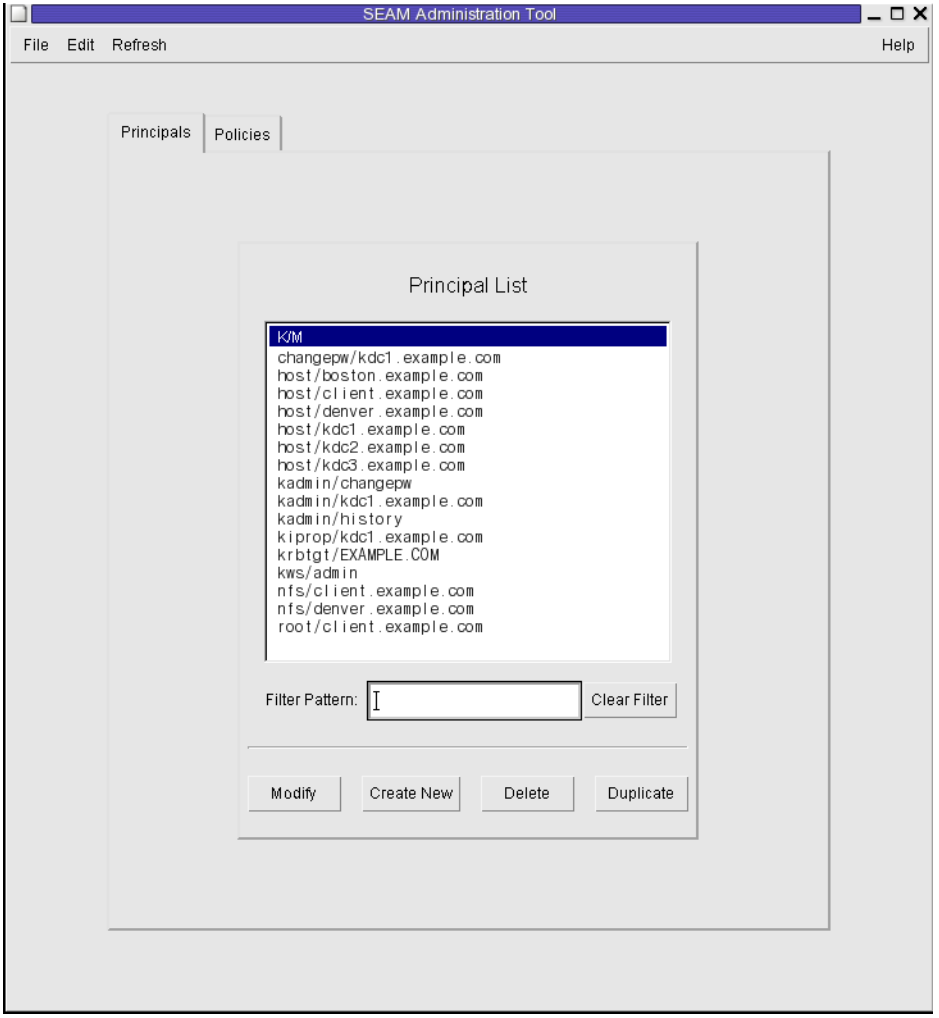
対応するコマンド行の例は、この手順のあとに示します。

1 必要に応じて、**SEAM** ツールを起動します。

詳細は、525 ページの「SEAM ツールを起動する方法」を参照してください。

```
$ /usr/sbin/gkadmin
```

2 「**Principals**」タブをクリックします。
主体の一覧が表示されます。



3 特定の主体を表示するか、主体の部分リストを表示します。
「Filter」フィールドにフィルタ文字列を入力して、Return キーを押します。フィルタが正常終了すると、フィルタに一致する主体の一覧が表示されます。

フィルタ文字列は、1 文字以上の文字列である必要があります。フィルタメカニズムでは大文字と小文字が区別されるため、大文字と小文字を正しく指定する必要があります。たとえば、フィルタ文字列に ge と入力すると、主体名に文字列 ge を含む主体 (george、 edge など) だけが表示されます。

すべての主体を表示するには、「Clear Filter」をクリックします。

例 25–1 Kerberos 主体の一覧の表示 (コマンド行)

次の例では、kadmin の list_principals コマンドを使用して、kadmin* と一致するすべての主体を表示します。list_principals コマンドでは、ワイルドカードを使用できます。

```
kadmin: list_principals kadmin*
kadmin/changepw@EXAMPLE.COM
kadmin/kdc1.example.con@EXAMPLE.COM
kadmin/history@EXAMPLE.COM
kadmin: quit
```

▼ Kerberos 主体の属性を表示する方法

対応するコマンド行の例は、この手順のあとに示します。

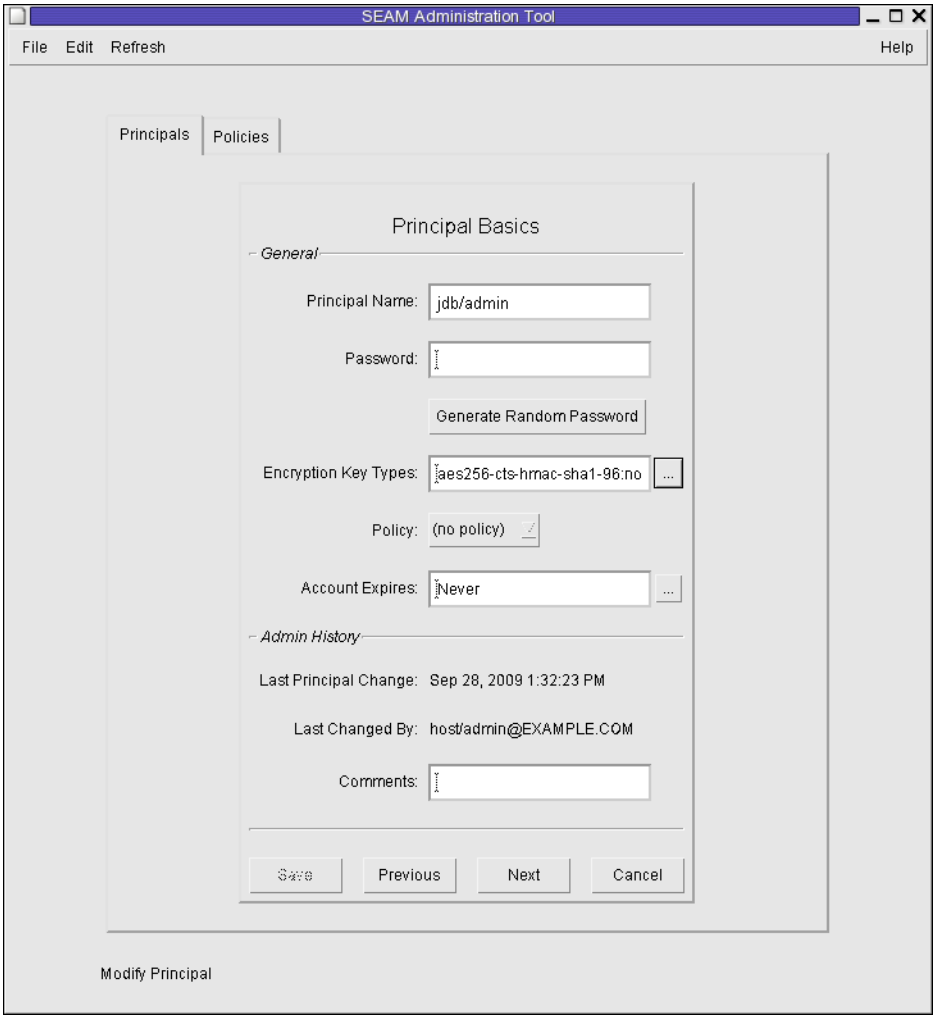
1. 必要に応じて、**SEAM** ツールを起動します。
   詳細は、525 ページの「SEAM ツールを起動する方法」を参照してください。

   ```
   $ /usr/sbin/gkadmin
   ```

2. 「**Principals**」タブをクリックします。
3. 表示する主体を一覧から選択して、「**Modify**」をクリックします。
   「Principal Basic」パネルが表示され、主体の属性の一部が示されます。
4. 「**Next**」をクリックして、主体のすべての属性を表示します。
   属性情報は、3 つのウィンドウに表示されます。「Help」メニューから「Context-Sensitive Help」を選択すると、各ウィンドウの属性に関する情報が表示されます。主体のすべての属性の説明については、547 ページの「SEAM ツールパネルの説明」を参照してください。
5. 表示を終了する場合は、「**Cancel**」をクリックします。

例 25–2 Kerberos 主体の属性の表示

次の例は、jdb/admin 主体を表示したときの最初のウィンドウです。



例 25–3 Kerberos 主体の属性の表示 (コマンド行)

次の例では、kadmin の get_principal コマンドを使用して、jdb/admin 主体の属性を表示します。

```
kadmin: getprinc jdb/admin
Principal: jdb/admin@EXAMPLE.COM

Expiration date: [never]
Last password change: [never]

Password expiration date: Wed Apr 14 11:53:10 PDT 2011
Maximum ticket life: 1 day 16:00:00
Maximum renewable life: 1 day 16:00:00
```

```
Last modified: Mon Sep 28 13:32:23 PST 2009 (host/admin@EXAMPLE.COM)
Last successful authentication: [never]
Last failed authentication: [never]
Failed password attempts: 0
Number of keys: 1
Key: vno 1, AES-256 CTS mode with 96-bit SHA-1 HMAC, no salt
Key: vno 1, AES-128 CTS mode with 96-bit SHA-1 HMAC, no salt
Key: vno 1, Triple DES with HMAC/sha1, no salt
Key: vno 1, ArcFour with HMAC/md5, no salt
Key: vno 1, DES cbc mode with RSA-MD5, no salt
Attributes: REQUIRES_HW_AUTH
Policy: [none]
kadmin: quit
```

▼ 新しい Kerberos 主体を作成する方法

対応するコマンド行の例は、この手順のあとに示します。

1 必要に応じて、**SEAM** ツールを起動します。
詳細は、525 ページの「SEAM ツールを起動する方法」を参照してください。

注 – 新しい主体を作成するときに、新しいポリシーが必要な場合は、新しいポリシーを作成してから新しい主体を作成する必要があります。543 ページの「新しい Kerberos ポリシーを作成する方法」を参照してください。

```
$ /usr/sbin/gkadmin
```

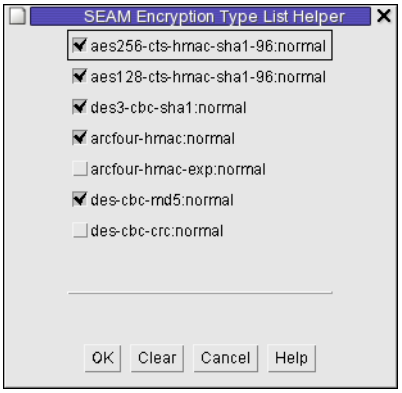
2 「**Principals**」タブをクリックします。

3 「**New**」をクリックします。
「Principal Basics」パネルが表示され、主体の属性の一部が示されます。

4 主体名とパスワードを指定します。
主体名とパスワードは必須です。

5 主体の暗号化タイプを指定します。

暗号化鍵タイプフィールドの右にあるボックスをクリックして、使用可能なすべての暗号化鍵タイプを表示する新しいウィンドウを開きます。必須の暗号化タイプを選択してから、「OK」をクリックします。



6 主体のポリシーを指定します。

7 主体の属性に値を指定します。「**Next**」をクリックして、属性の値を必要に応じて指定します。

属性情報は、3 つのウィンドウに表示されます。「Help」メニューから「Context-Sensitive Help」を選択すると、各ウィンドウの属性に関する情報が表示されます。主体のすべての属性の説明については、547 ページの「SEAM ツールパネルの説明」を参照してください。

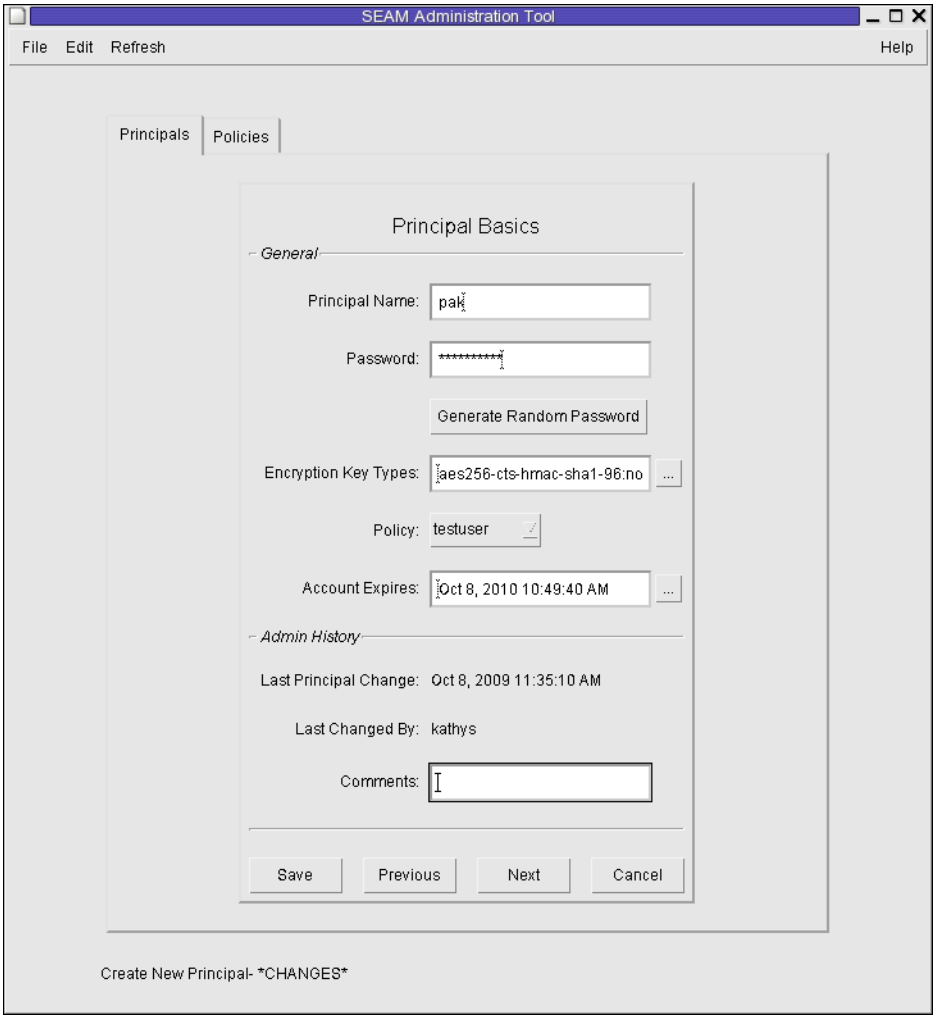
8 主体を保存する場合は、「**Save**」をクリックします。または、最後のパネルで「**Done**」をクリックします。

9 必要に応じて、新しい主体の **Kerberos** 管理権限を **`/etc/krb5/kadm5.acl`** ファイルに設定します。

詳細は、537 ページの「Kerberos 管理権限を変更する方法」を参照してください。

例 25–4 新しい Kerberos 主体の作成

次の例は、`pak` という新しい主体を作成するときの「Principal Basics」パネルです。ポリシーには、`testuser` が設定されています。



例 25–5 新しい Kerberos 主体の作成 (コマンド行)

次の例では、kadmin の add_principal コマンドを使用して、 pak という新しい主体を作成します。この主体のポリシーには、testuser が設定されています。

```
kadmin: add_principal -policy testuser pak
Enter password for principal "pak@EXAMPLE.COM": <Type the password>
Re-enter password for principal "pak@EXAMPLE.COM": <Type the password again>
Principal "pak@EXAMPLE.COM" created.
kadmin: quit
```

▼ Kerberos 主体を複製する方法

この手順では、既存の主体の一部またはすべてを使用して、新しい主体を作成する方法について説明します。この手順に対応するコマンド行はありません。

1 必要に応じて、**SEAM** ツールを起動します。

詳細は、525 ページの「SEAM ツールを起動する方法」を参照してください。

```
$ /usr/sbin/gkadmin
```

2 「**Principals**」タブをクリックします。

3 複製する主体を一覧から選択して、「**Duplicate**」をクリックします。

「Principal Basics」パネルが表示されます。選択した主体のすべての属性が複製されます。ただし、「Principal Name」と「Password」フィールドは複製されず、空で表示されます。

4 主体名とパスワードを指定します。

主体名とパスワードは必須です。選択した主体をそのまま複製するときは、「Save」をクリックして、手順 7 に進みます。

5 主体の属性に別の値を指定します。「**Next**」をクリックして、属性の値を必要に応じて指定します。

属性情報は、3 つのウィンドウに表示されます。「Help」メニューから「Context-Sensitive Help」を選択すると、各ウィンドウの属性に関する情報が表示されます。主体のすべての属性の説明については、547 ページの「SEAM ツールパネルの説明」を参照してください。

6 主体を保存する場合は、「**Save**」をクリックします。または、最後のパネルで「**Done**」をクリックします。

7 必要に応じて、主体の **Kerberos** 管理権限を **/etc/krb5/kadm5.acl** ファイルに設定します。

詳細は、537 ページの「Kerberos 管理権限を変更する方法」を参照してください。

▼ Kerberos 主体を変更する方法

対応するコマンド行の例は、この手順のあとに示します。

1 必要に応じて、**SEAM** ツールを起動します。

詳細は、525 ページの「SEAM ツールを起動する方法」を参照してください。

```
$ /usr/sbin/gkadmin
```

2 「**Principals**」タブをクリックします。

3 変更する主体を一覧から選択して、「**Modify**」をクリックします。
「Principal Basic」パネルが表示され、主体の属性の一部が示されます。

4 主体の属性を変更します。「**Next**」をクリックして、必要に応じて属性を変更します。
属性情報は、3 つのウィンドウに表示されます。「Help」メニューから「Context-Sensitive Help」を選択すると、各ウィンドウの属性に関する情報が表示されます。主体のすべての属性の説明については、547 ページの「SEAM ツールパネルの説明」を参照してください。

注 – 主体名は変更できません。主体名を変更するときは、主体を複製し、新しい名前を指定して保存してから、古い主体を削除する必要があります。

---

5 主体を保存する場合は、「**Save**」をクリックします。または、最後のパネルで「**Done**」をクリックします。

6 **/etc/krb5/kadm5.acl** ファイルで、主体の **Kerberos** 管理権限を変更します。
詳細は、537 ページの「Kerberos 管理権限を変更する方法」を参照してください。

例 25–6 Kerberos 主体のパスワードの変更 (コマンド行)

次の例では、`kadmin` の `change_password` コマンドを使用して、`jdb` 主体のパスワードを変更します。`change_password` コマンドでは、主体のパスワード履歴に存在するパスワードには変更できません。

```
kadmin: change_password jdb
Enter password for principal "jdb": <Type the new password>
Re-enter password for principal "jdb": <Type the password again>
Password for "jdb@EXAMPLE.COM" changed.
kadmin: quit
```

主体のその他の属性を変更するには、`kadmin` の `modify_principal` コマンドを使用する必要があります。

▼ Kerberos 主体を削除する方法

対応するコマンド行の例は、この手順のあとに示します。

1 必要に応じて、**SEAM** ツールを起動します。
詳細は、525 ページの「SEAM ツールを起動する方法」を参照してください。

```
$ /usr/sbin/gkadmin
```

2 「**Principals**」タブをクリックします。

3 削除する主体を一覧から選択して、「**Delete**」をクリックします。
削除を確定すると、主体が削除されます。

4 **Kerberos** アクセス制御リスト **(ACL)** ファイル **/etc/krb5/kadm5.acl** から主体を削除します。
詳細は、537 ページの「Kerberos 管理権限を変更する方法」を参照してください。

例 25–7 Kerberos 主体の削除 (コマンド行)

次の例では、`kadmin` の `delete_principal` コマンドを使用して、`jdb` 主体を削除します。

```
kadmin: delete_principal pak
Are you sure you want to delete the principal "pak@EXAMPLE.COM"? (yes/no): yes
Principal "pak@EXAMPLE.COM" deleted.
Make sure that you have removed this principal from all ACLs before reusing.
kadmin: quit
```

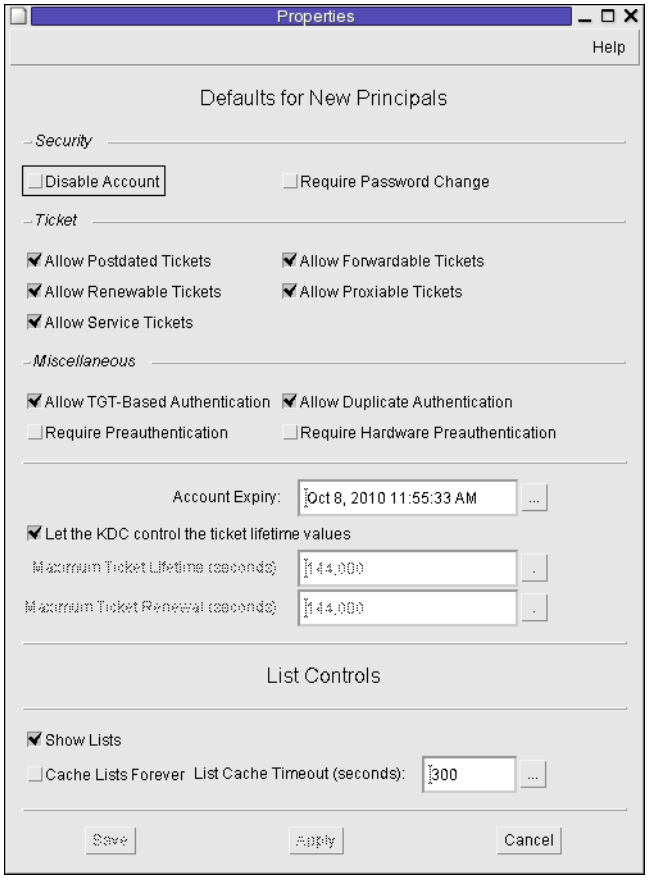
▼ 新しい Kerberos 主体を作成するときのデフォルトを設定する方法

この手順に対応するコマンド行はありません。

1 必要に応じて、**SEAM** ツールを起動します。
詳細は、525 ページの「SEAM ツールを起動する方法」を参照してください。

```
$ /usr/sbin/gkadmin
```

2 「**Edit**」メニューから「**Properties**」を選択します。
「Properties」ウィンドウが表示されます。



3 新しい主体を作成するときに使用するデフォルトを選択します。
「Help」メニューから「Context-Sensitive Help」を選択すると、各ウィンドウの属性に関する情報が表示されます。

4 「**Save**」をクリックします。

▼ Kerberos 管理権限を変更する方法

使用する環境には、多くのユーザー主体が登録されていると思われます。しかし、Kerberos データベースの管理者は通常、少数のユーザーだけに割り当てます。Kerberos データベースを管理する権限は、Kerberos アクセス制御リスト (ACL)

ファイル (kadm5.acl) によって判断されます。kadm5.acl ファイルを使用すると、主体ごとに権限を設定できます。主体名にワイルドカード (*) を使用すると、複数の主体に権限を指定できます。

1 マスター KDC 上でスーパーユーザーになります。

2 **/etc/krb5/kadm5.acl** ファイルを編集します。

kadm5.acl ファイルのエントリは、次の書式で記述する必要があります。

principal privileges [*principal-target*]

principal 権限を与える主体を指定します。主体名の任意の場所にワイルドカード (*) を使用できます。複数の主体グループに同じ権限を与えるときに使用します。たとえば、admin インスタンスを持つすべての主体を指定する場合は、*/admin@*realm* を使用します。

admin インスタンスは通常、個別の権限 (Kerberos データベースへの管理アクセス権など) を個別の Kerberos 主体に許可するときに使用します。たとえば、ユーザー jdb が、jdb/admin という管理目的の主体を持つとします。この場合、ユーザー jdb は、この権限を実際に使用するときにだけ、jdb/admin チケットを取得します。

privileges 主体が実行できる操作または実行できない操作を指定します。このフィールドは、次に示す文字のリストのうち 1 つまたは複数の文字列 (またはその大文字) で構成されます。文字が大文字 (または指定されない) 場合、その操作は許可されません。文字が小文字の場合、その操作は許可されます。

a	主体またはポリシーの追加を許可します/しません。
d	主体またはポリシーの削除を許可します/しません。
m	主体またはポリシーの変更を許可します/しません。
c	主体のパスワードの変更を許可します/しません。
i	Kerberos データベースの照会を許可します/しません。
l	Kerberos データベース内の主体またはポリシーの一覧表示を許可します/しません。
x または *	すべての権限 (admcil) を許可します。

principal-target このフィールドに主体を指定すると、*principal* の操作が *principal-target* の場合にだけ、*privileges* が *principal* に適用されます。主体名の任意の場所にワイルドカード (*) を使用できます。主体をグループ化するときに使用します。

例 25–8 Kerberos 管理権限の変更

kadm5.acl ファイル内の次のエントリは、EXAMPLE.COM レルム内で admin インスタンスを持つ すべての主体に対して、Kerberos データベース上のすべての権限を与えます。

```
*/admin@EXAMPLE.COM *
```

kadm5.acl ファイル内の次のエントリは、jdb@EXAMPLE.COM 主体に対して、root インスタンスを持つすべての主体に関する追加、一覧表示、および照会の権限を与えます。

```
jdb@EXAMPLE.COM ali */root@EXAMPLE.COM
```

Kerberos 主体の管理

この節では、SEAM ツールを使用してポリシーを管理する手順について説明します。また、対応するコマンド行がある場合は、その例も示します。

Kerberos ポリシーの管理 (作業マップ)

作業	説明	参照先
ポリシーの一覧を表示します。	「Policies」タブをクリックして、ポリシーの一覧を表示します。	540 ページの「Kerberos ポリシーの一覧を表示する方法」
ポリシーの属性を表示します。	「Policy List」からポリシーを選択し、「Modify」ボタンをクリックして、ポリシーの属性を表示します。	541 ページの「Kerberos ポリシーの属性を表示する方法」
新しいポリシーを作成します。	「Policy List」パネルの「Create New」ボタンをクリックして、新しいポリシーを作成します。	543 ページの「新しい Kerberos ポリシーを作成する方法」
ポリシーを複製します。	複製するポリシーを「Policy List」ポリシーから選択し、「Duplicate」ボタンをクリックして、ポリシーを複製します。	545 ページの「Kerberos ポリシーを複製する方法」
ポリシーを変更します。	変更するポリシーを「Policy List」ポリシーから選択し、「Modify」ボタンをクリックして、ポリシーを変更します。 ポリシー名は変更できません。ポリシー名を変更するときは、ポリシーを複製し、新しい名前を指定して保存してから、古いポリシーを削除する必要があります。	545 ページの「Kerberos ポリシーを変更する方法」
ポリシーを削除します。	削除するポリシーを「Policy List」ポリシーから選択し、「Delete」ボタンをクリックして、ポリシーを削除します。	546 ページの「Kerberos ポリシーを削除する方法」

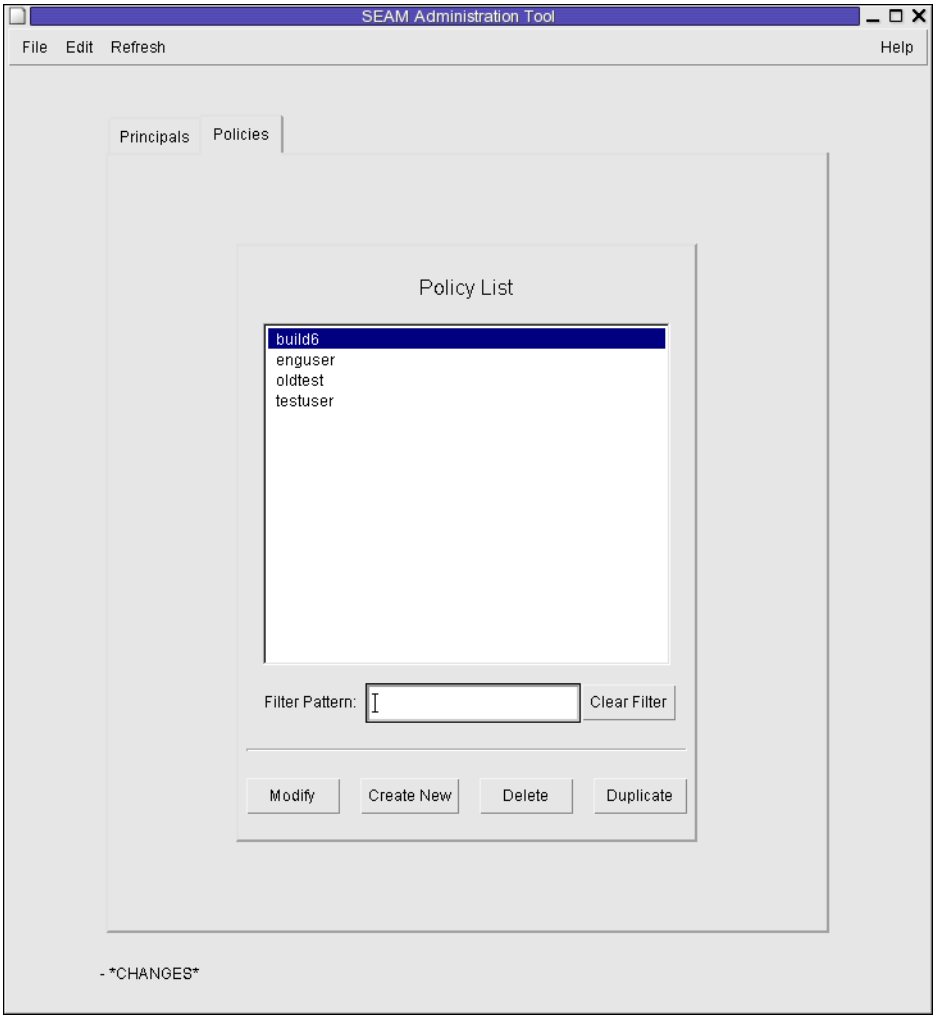
▼ Kerberos ポリシーの一覧を表示する方法

対応するコマンド行の例は、この手順のあとに示します。

1 必要に応じて、**SEAM** ツールを起動します。
詳細は、525 ページの「SEAM ツールを起動する方法」を参照してください。

```
$ /usr/sbin/gkadmin
```

2 「**Policies**」タブをクリックします。
ポリシーの一覧が表示されます。



3 特定のポリシーを表示するか、ポリシーの部分リストを表示します。

「Filter」フィールドにフィルタ文字列を入力して、Return キーを押します。フィルタが正常終了すると、フィルタに一致するポリシーの一覧が表示されます。

フィルタ文字列は、1 文字以上の文字列である必要があります。フィルタメカニズムでは大文字と小文字が区別されるため、大文字と小文字を正しく指定する必要があります。たとえば、フィルタ文字列に `ge` と入力すると、ポリシー名に文字列 `ge` を含むポリシー (`george`、 `edge` など) だけが表示されます。

すべてのポリシーを表示するには、「Clear Filter」をクリックします。

例 25–9 Kerberos ポリシーの一覧の表示 (コマンド行)

次の例では、`kadmin` の `list_policies` コマンドを使用して、`*user*` と一致するすべてのポリシーを表示します。`list_policies` コマンドでは、ワイルドカードを使用できます。

```
kadmin: list_policies *user*
testuser
enguser
kadmin: quit
```

▼ Kerberos ポリシーの属性を表示する方法

対応するコマンド行の例は、この手順のあとに示します。

1 必要に応じて、**SEAM** ツールを起動します。

詳細は、525 ページの「SEAM ツールを起動する方法」を参照してください。

```
$ /usr/sbin/gkadmin
```

2 「**Policies**」タブをクリックします。

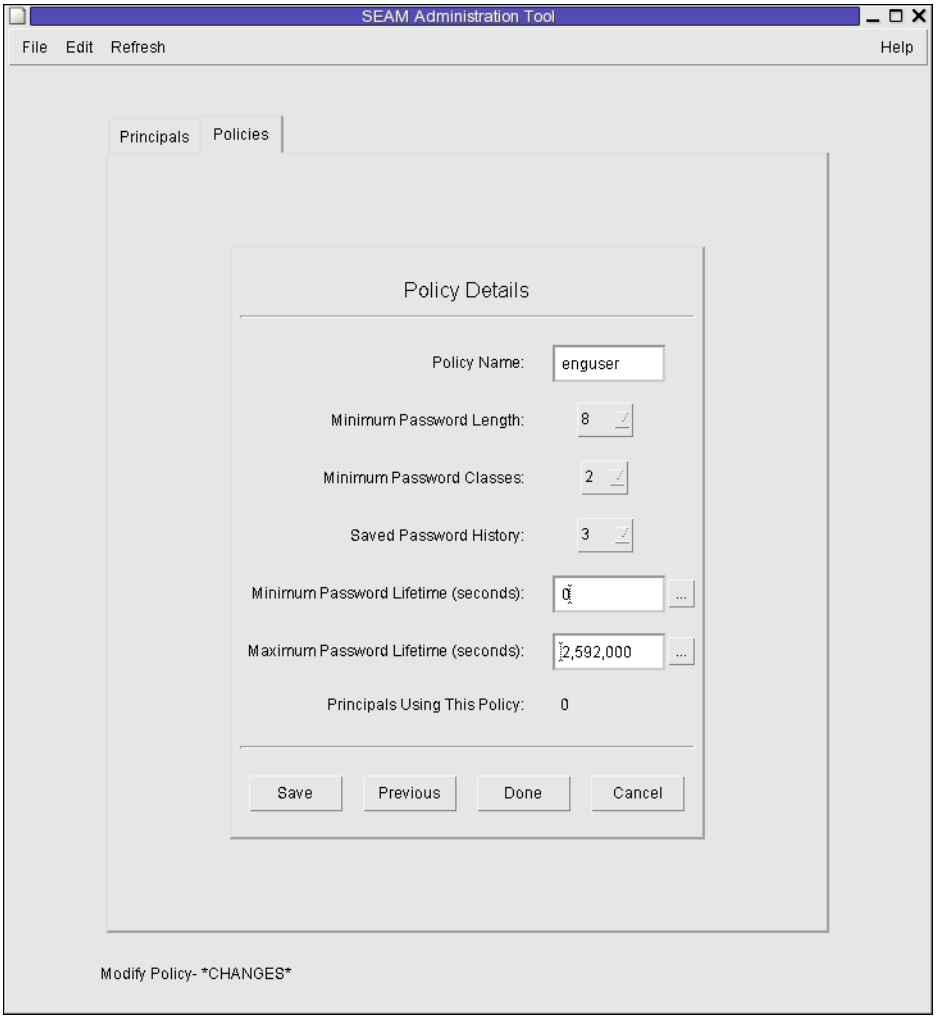
3 表示するポリシーを一覧から選択して、「**Modify**」をクリックします。

「Policy Details」パネルが表示されます。

4 表示を終了する場合は、「**Cancel**」をクリックします。

例 25–10 Kerberos ポリシーの属性の表示

次の例は、`test` ポリシーを表示したときの「Policy Details」パネルです。



例 25–11 Kerberos ポリシーの属性の表示 (コマンド行)

次の例では、kadmin の get_policy コマンドを使用して、enguser ポリシーの属性を表示します。

```
kadmin: get_policy enguser
Policy: enguser
Maximum password life: 2592000
Minimum password life: 0
Minimum password length: 8
Minimum number of password character classes: 2
Number of old keys kept: 3
Reference count: 0
kadmin: quit
```

「Reference count」は、このポリシーを使用する主体の数です。

▼ 新しい Kerberos ポリシーを作成する方法

対応するコマンド行の例は、この手順のあとに示します。

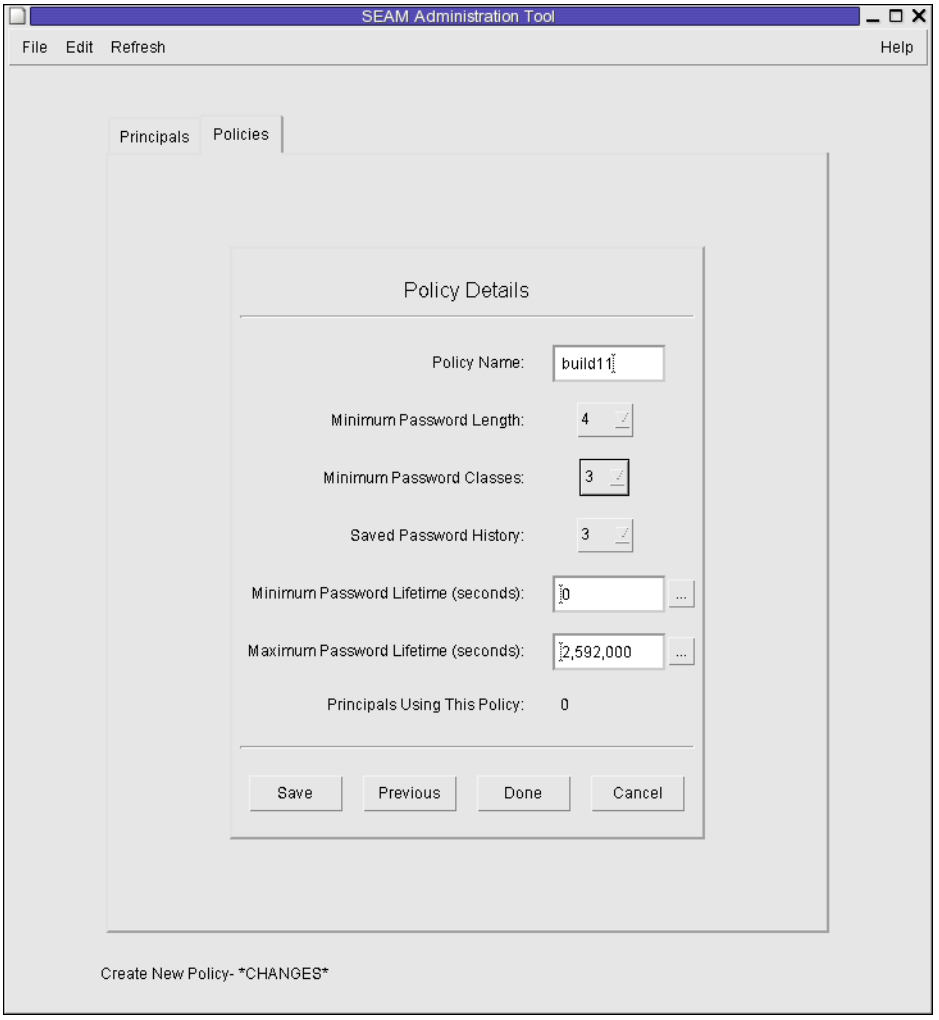
1. 必要に応じて、**SEAM** ツールを起動します。
   詳細は、525 ページの「SEAM ツールを起動する方法」を参照してください。

   ```
   $ /usr/sbin/gkadmin
   ```

2. 「**Policies**」タブをクリックします。
3. 「**New**」をクリックします。
   「Policy Details」パネルが表示されます。
4. 「**Policy Name**」フィールドにポリシー名を指定します。
   ポリシー名は必須です。
5. ポリシーの属性の値を指定します。
   「Help」メニューから「Context-Sensitive Help」を選択すると、このウィンドウの属性に関する情報が表示されます。あるいは、ポリシーのすべての属性の説明については、表 25–5 を参照してください。
6. 「**Save**」をクリックしてポリシーを保存するか、「**Done**」をクリックします。

例 25–12 新しい Kerberos ポリシーの作成

次の例では、`build11` という新しいポリシーを作成します。「Minimum Password Classes」は、3 に設定されています。



例 25–13 新しい Kerberos ポリシーの作成 (コマンド行)

次の例では、kadmin の add_policy コマンドを使用して、build11 ポリシーを作成します。このポリシーのパスワードには、3 種類以上の文字クラスが必要です。

```
$ kadmin
kadmin: add_policy -minclasses 3 build11
kadmin: quit
```

▼ Kerberos ポリシーを複製する方法

この手順では、既存のポリシーの一部またはすべてを使用して、新しいポリシーを作成する方法について説明します。この手順に対応するコマンド行はありません。

1 必要に応じて、**SEAM** ツールを起動します。

詳細は、525 ページの「SEAM ツールを起動する方法」を参照してください。

```
$ /usr/sbin/gkadmin
```

2 「**Policies**」タブをクリックします。

3 複製するポリシーを一覧から選択して、「**Duplicate**」をクリックします。

「Policy Details」パネルが表示されます。選択したフィールドのすべての属性が複製されます。ただし、「Policy Name」フィールドは空で表示されます。

4 複製するポリシー名を「**Policy Name**」フィールドに指定します。

ポリシー名は必須です。選択したポリシーをそのまま複製するには、手順 6 に進みます。

5 ポリシーの属性に別の値を指定します。

「Help」メニューから「Context-Sensitive Help」を選択すると、このウィンドウの属性に関する情報が表示されます。あるいは、ポリシーのすべての属性の説明については、表 25–5 を参照してください。

6 「**Save**」をクリックしてポリシーを保存するか、「**Done**」をクリックします。

▼ Kerberos ポリシーを変更する方法

対応するコマンド行の例は、この手順のあとに示します。

1 必要に応じて、**SEAM** ツールを起動します。

詳細は、525 ページの「SEAM ツールを起動する方法」を参照してください。

```
$ /usr/sbin/gkadmin
```

2 「**Policies**」タブをクリックします。

3 変更するポリシーを一覧から選択して「**Modify**」をクリックします。

「Policy Details」パネルが表示されます。

4 ポリシーの属性を変更します。

「Help」メニューから「Context-Sensitive Help」を選択すると、このウィンドウの属性に関する情報が表示されます。あるいは、ポリシーのすべての属性の説明については、表 25–5 を参照してください。

注 – ポリシー名は変更できません。ポリシー名を変更するときは、ポリシーを複製し、新しい名前を指定して保存してから、古いポリシーを削除する必要があります。

---

5 「**Save**」をクリックしてポリシーを保存するか、「**Done**」をクリックします。

例 **25–14** Kerberos ポリシーの変更 (コマンド行)

次の例では、`kadmin` の `modify_policy` コマンドを使用して、`build11` ポリシーの最小パスワード長を 5 文字に変更します。

```
$ kadmin
kadmin: modify_policy -minlength 5 build11
kadmin: quit
```

▼ Kerberos ポリシーを削除する方法

対応するコマンド行の例は、この手順のあとに示します。

注 – ポリシーを削除する前に、現在使用しているすべての主体からそのポリシーを取り消す必要があります。ポリシーを取り消すには、その主体の「Policy」属性を変更する必要があります。任意の主体が使用しているポリシーは、削除できません。

---

1 必要に応じて、**SEAM** ツールを起動します。

詳細は、525 ページの「SEAM ツールを起動する方法」を参照してください。

```
$ /usr/sbin/gkadmin
```

2 「**Policies**」タブをクリックします。

3 削除するポリシーを一覧から選択して、「**Delete**」をクリックします。

削除を確定すると、ポリシーが削除されます。

例 **25–15** Kerberos ポリシーの削除 (コマンド行)

次の例では、`kadmin` の `delete_policy` コマンドを使用して、`build11` ポリシーを削除します。

```
kadmin: delete_policy build11
Are you sure you want to delete the policy "build11"? (yes/no): yes
kadmin: quit
```

ポリシーを削除する前に、現在使用しているすべての主体からそのポリシーを取り消す必要があります。ポリシーを取り消すには、関係する主体に対して `kadmin` の `modify_principal -policy` コマンドを使用する必要があります。そのポリシーが主体に使用されている場合は、`delete_policy` コマンドは失敗します。

SEAM ツール参照

この節では、SEAM ツールの各パネルについて説明します。SEAM ツールで制限された権限を使用する方法についても説明します。

SEAM ツールパネルの説明

この節では、SEAM ツールで指定または表示できる主体とポリシーの属性について説明します。属性は、表示されるパネルごとに分類されています。

表 25–2　SEAM ツールの「Principal Basics」パネルの属性

属性	説明
Principal Name	主体名 (完全指定形式の主体名の *primary*/*instance* 部分)。主体は、KDC がチケットを割り当てることができる一意の ID です。 主体を変更しても、主体名は編集できません。
パスワード	主体のパスワード。「Generate Random Password」ボタンを使用して、主体のランダムパスワードを作成できます。
Policy	主体に使用できるポリシーのメニュー。
Account Expires	主体のアカウントが期限切れになる日時。アカウントが期限切れになると、主体はチケット認可チケット (TGT) を取得できず、ログインできなくなります。
Last Principal Change	主体の情報が最後に変更された日付。(読み取り専用)
Last Changed By	この主体のアカウントを最後に変更した主体名。(読み取り専用)
Comments	主体に関連するコメント (「一時アカウント」など)。

表 25–3　SEAM ツールの「Principal Details」パネルの属性

属性	説明
Last Success	主体が最後に正常にログインした日時。(読み取り専用)
Last Failure	主体が最後にログインに失敗した日時。(読み取り専用)

表 25–3 SEAM ツールの「Principal Details」パネルの属性 (続き)

属性	説明
Failure Count	主体のログインが失敗した回数。(読み取り専用)
Last Password Change	主体のパスワードが最後に変更された日時。(読み取り専用)
Password Expires	主体の現在のパスワードが期限切れになる日時。
Key Version	主体の鍵のバージョン番号。この属性は通常、パスワードが危険にさらされた場合にだけ変更されます。
Maximum Lifetime (seconds)	チケットを主体が使用できる最大期間 (更新しない場合)。
Maximum Renewal (seconds)	既存のチケットを主体が更新できる最大期間。

表 25–4 SEAM ツールの「Principal Flags」パネルの属性

属性 (ラジオボタン)	説明
Disable Account	チェックすると、その主体はログインできなくなります。この属性は、主体のアカウントを一時的に凍結するときに使用します。
Require Password Change	チェックすると、主体の現在のパスワードが期限切れとなり、ユーザーは `kpasswd` コマンドを使用して新しいパスワードを作成しなければなりません。この属性は、セキュリティー侵害が発生し、古いパスワードを置換する必要があるときに使用します。
Allow Postdated Tickets	チェックすると、主体は遅延チケットを取得できます。 たとえば、`cron` ジョブを数時間後に実行する場合は、遅延チケットを使用する必要があります。ただし、チケットの有効期限が短い場合は、事前にチケットを取得できません。
Allow Forwardable Tickets	チェックすると、主体は転送可能チケットを取得できます。 転送可能チケットは、遠隔ホストに転送されて、シングルサインオンセッションを実現します。たとえば、転送可能チケットを使用して、ユーザー自身の `ftp` 認証または `rsh` 認証が完了すると、NFS サービスなどのほかのサービスを利用するときに、新たにパスワードを要求されません。
Allow Renewable Tickets	チェックすると、主体が更新可能チケットを取得できます。 主体は、チケットが更新可能な場合、有効期限日時を自動的に延長することができます。つまり、最初のチケットの期限が切れても、新しいチケットを取得する必要がありません。現在の NFS サービスは、チケットを新しくするチケットサービスです。
Allow Proxiable Tickets	チェックすると、主体は代理可能チケットを取得できます。 代理可能チケットを使用すると、クライアントの代わりにサービスがクライアントの操作を実行できます。代理可能チケットを使用すると、サービスはクライアントの ID を使用して別のサービスのチケットを取得できます。ただし、チケット認可チケット (TGT) を取得することはできません。

表 25–4 SEAM ツールの「Principal Flags」パネルの属性 (続き)

属性 (ラジオボタン)	説明
Allow Service Tickets	チェックすると、サービスチケットを特定の主体に発行できます。 サービスチケットの発行は、`kadmin/`*hostname* および `changepw/`*hostname* 主体に許可してはいけません。これらの主体は、KDC データベース以外は更新してはいけません。
Allow TGT-Based Authentication	チェックすると、このサービス主体は別の主体にサービスを提供できます。つまり、KDC は、サービス主体にサービスチケットを発行できます。 この属性は、サービス主体にだけ使用できます。チェックを解除すると、サービスチケットをサービス主体に対して発行できません。
Allow Duplicate Authentication	チェックすると、このユーザー主体はほかのユーザー主体のサービスチケットを取得できます。 この属性は、ユーザー主体にだけ使用できます。チェックを解除すると、ユーザー主体はサービス主体のサービスチケットを取得できますが、ほかのユーザー主体のサービスチケットは取得できません。
Required Preauthentication	チェックすると、KDC が要求されたチケット認可チケット (TGT) を主体に送信する前に、その主体が TGT を要求している主体であることを KDC のソフトウェアが認証します。この事前認証は通常、DES カードなどの特別のパスワードを使用して行われます。 チェックを解除すると、KDC は要求された TGT を主体に送信する前に、主体の事前認証を必要としません。
Required Hardware Authentication	チェックすると、KDC が要求されたチケット認可チケット (TGT) を主体に送信する前に、その主体が TGT を要求している主体であることを KDC のハードウェアが認証します。ハードウェア事前認証は、たとえば Java リングのリーダー上で行われます。 チェックを解除すると、KDC は要求された TGT を主体に送信する前に、主体の事前認証を必要としません。

表 25–5 SEAM ツールの「Policy Basics」区画の属性

属性	説明
ポリシー名	ポリシー名。ポリシーとは、主体のパスワードとチケットを管理する一連のルールのことです。 ポリシーを変更しても、ポリシー名は編集できません。
Minimum Password Length	主体の最小パスワード長。

表 25–5 SEAM ツールの「Policy Basics」区画の属性 (続き)

属性	説明
Minimum Password Classes	主体のパスワードに必要な異なる文字タイプの数。 たとえば、最小クラス値が 2 の場合は、パスワードに 2 種類以上の文字タイプを使用する必要があります。たとえば、英字と数字を使用して「hi2mom」と入力する必要があります。値が 3 の場合は、パスワードに 3 種類以上の文字タイプを使用する必要があります。たとえば、英字、数字、および句読点を使用して「hi2mom!」と入力する必要があります。 値が 1 の場合は、パスワード文字タイプの数に制限が設定されません。
Saved Password History	主体に使用された過去のパスワードの数と、過去のパスワードの一覧。これらのパスワードは再使用できません。
Minimum Password Lifetime (seconds)	パスワードの最小使用期間。この期間が経過しないとパスワードを変更できません。
Maximum Password Lifetime (seconds)	パスワードの最大使用期間。この期間が経過したらパスワードを変更する必要があります。
Principals Using This Policy	このポリシーが現在適用されている主体の数。(読み取り専用)

Kerberos 管理権限を制限して SEAM ツールを使用する

admin 主体が Kerberos データベースの管理権限をすべて持っている場合は、SEAM ツールの機能をすべて使用できます。ただし、たとえば主体の一覧の表示、主体のパスワードの変更だけができるように、Kerberos 管理権限を制限することもできます。Kerberos 管理権限を制限した場合でも、SEAM ツールを使用できます。ただし、許可された Kerberos 管理権限によって、SEAM ツールの使い方が異なります。表 25–6 は、Kerberos 管理権限に基づいた SEAM ツールの変更の一覧です。

一覧表示権限がない場合、SEAM ツールの表示がもっとも大きく変わります。この場合、操作する主体とポリシーの一覧が「List」パネルに表示されません。代わりに、「List」パネルの「Name」フィールドを使用して、操作する主体またはポリシーを指定する必要があります。

SEAM ツールにログインしても、必要な権限がない場合は、次のメッセージが表示されて「SEAM Administration Login」ウィンドウに戻ります。

```
Insufficient privileges to use gkadmin: ADMCIL. Please try using another principal.
```

主体が Kerberos データベースを管理する権限を変更する方法については、537 ページの「Kerberos 管理権限を変更する方法」を参照してください。

表 25–6　Kerberos 管理権限が制限された SEAM ツールの使用

許可しない権限	SEAM ツールの変更
a (追加)	「Principal List」および「Policy List」パネルの「Create New」と「Duplicate」ボタンを使用できません。追加権限がない場合は、新しい主体またはポリシーを作成または複製できません。
d (削除)	「Principal List」および「Policy List」パネルの「Delete」ボタンを使用できません。削除権限がない場合は、主体またはポリシーを削除できません。
m (変更)	「Principal List」および「Policy List」パネルの「Modify」ボタンを使用できません。変更権限がない場合は、主体またはポリシーを変更できません。 また、「Modify」ボタンを使用できない場合、パスワードの変更権限を持っていても、主体のパスワードを変更できません。
c (パスワードの変更)	「Principal Basics」パネルの「Password」フィールドが読み取り専用になり、変更できません。パスワードの変更権限がない場合、主体のパスワードを変更できません。 パスワードの変更権限を持っている場合でも、主体のパスワードを変更するときは、さらに変更権限が必要になります。
i (データベースの照会)	「Principal List」および「Policy List」パネルの「Modify」と「Duplicate」ボタンを使用できません。照会権限がない場合は、主体またはポリシーを変更または複製できません。 また、「Modify」ボタンを使用できない場合、パスワードの変更権限を持っていても、主体のパスワードを変更できません。
l (一覧)	「List」パネルで主体とポリシーの一覧を表示できません。一覧権限がない場合は、「List」パネルの「Name」フィールドを使用して、操作する主体またはポリシーを指定する必要があります。

キータブファイルの管理

サービスを提供するすべてのホストには、「キータブ」(「鍵テーブル」の短縮名)と呼ばれるローカルファイルが必要です。キータブには、「サービス鍵」と呼ばれる該当するサービスの主体が格納されます。サービス鍵は、KDC に対してサービス自身を認証するときに使用され、Kerberos とそのサービスだけが認識します。たとえば、Kerberos NFS サーバーを使用する場合、このサーバーには `nfs` サービス主体を含むキータブが必要です。

キータブファイルにサービス鍵を追加するには、kadmin の ktadd コマンドを使用して、適切なサービス主体をホストのキータブファイルに追加します。サービス主体をキータブファイルに追加するときは、Kerberos データベースにあらかじめ主体を登録し、kadmin が主体の存在を検証できるようにする必要があります。マスター KDC では、キータブファイルのデフォルトの位置は /etc/krb5/kadm5.keytab です。Kerberos サービスを提供するアプリケーションサーバーでは、キータブファイルのデフォルトの位置は /etc/krb5/krb5.keytab です。

キータブはユーザーのパスワードに似ています。ユーザーの場合は、自分のパスワードを保護することが重要ですが、アプリケーションサーバーの場合は、キータブファイルを保護することが重要です。キータブファイルは常時ローカルディスクに格納し、root ユーザー以外は読み取れないようにしてください。また、キータブファイルは、セキュリティー保護されていないネットワークを介して送信しないでください。

root 主体をホストのキータブファイルに追加する特別な場合もあります。Kerberos クライアント上のユーザーが root と同等のアクセスを必要とする Kerberos NFS ファイルシステムをマウントするようにするには、クライアントの root 主体をクライアントのキータブファイルに追加する必要があります。追加しなかった場合、root アクセスで Kerberos NFS ファイルシステムをマウントするたびに、ユーザーは kinit コマンドをスーパーユーザーとして使用して、クライアントの root 主体の資格を取得する必要があります。これは、オートマウンタを使用している場合でも同様です。

注 – マスター KDC を設定するときは、kadmind および changepw 主体を kadm5.keytab ファイルに追加する必要があります。

キータブファイルを管理するときに、ktutil コマンドも使用できます。この対話型のコマンドは、Kerberos 管理権限がなくても、ローカルホストのキータブファイルを管理できます。kadmin は Kerberos データベースと対話しますが、ktutil は対話しないためです。つまり、主体をキータブファイルに追加したあとに ktutil を使用すると、キータブファイル内のキー一覧を表示したり、サービスの認証を一時的に無効にしたりできます。

注 – kadmin の ktadd コマンドを使用してキータブファイル内の主体を変更すると、新しい鍵が生成され、キータブファイルに追加されます。

キータブファイルの管理 (作業マップ)

作業	説明	参照先
サービス主体をキータブファイルに追加します。	kadmin の ktadd コマンドを使用して、サービス主体をキータブファイルに追加します。	553 ページの「Kerberos サービス主体をキータブファイルに追加する方法」
キータブファイルからサービス主体を削除します。	kadmin の ktremove コマンドを使用して、キータブファイルからサービスを削除します。	555 ページの「キータブファイルからサービス主体を削除する方法」
キータブファイル内のキー一覧 (主体の一覧) を表示します。	ktutil コマンドを使用して、キータブファイル内のキー一覧を表示します。	556 ページの「キータブファイル内のキー一覧 (主体) を表示する方法」
ホスト上でのサービスの認証を一時的に無効にします。	この手順を行うと、kadmin 権限がなくても、ホスト上でのサービスの認証を一時的に無効にできます。 ktutil を使用してサーバーのキータブファイルからサービス主体を削除する前に、元のキータブファイルを一時的な位置にコピーする必要があります。サービスを再度有効にする場合は、元のキータブファイルを適切な場所に戻す必要があります。	557 ページの「ホスト上のサービスの認証を一時的に無効にする方法」

▼ Kerberos サービス主体をキータブファイルに追加する方法

1. 主体がすでに Kerberos データベースに登録されていることを確認します。
   詳細は、527 ページの「Kerberos 主体の一覧を表示する方法」を参照してください。
2. キータブファイルに主体を追加するホスト上でスーパーユーザーになります。
3. **kadmin** コマンドを起動します。

   ```
   # /usr/sbin/kadmin
   ```

4. **ktadd** コマンドを使用して、キータブファイルに主体を追加します。

   kadmin: **ktadd** [**-e** *enctype*] [**-k** *keytab*] [**-q**] [*principal* | **-glob** *principal-exp*]

   -e *enctype*　krb5.conf ファイルに定義された暗号化タイプの一覧を上書きします。

   -k *keytab*　キータブファイルを指定します。デフォルトでは、/etc/krb5/krb5.keytab が使用されます。

`-q`	冗長な情報を表示しません。
principal	キータブファイルに追加する主体を指定します。追加できるサービス主体は、次のとおりです。 `host`、 `root`、 `nfs`、 および `ftp`。
`-glob` *principal-exp*	主体表現を指定します。*principal-exp* に一致するすべての主体が、キータブファイルに追加されます。主体表現の規則は、`kadmin` の `list_principals` コマンドと同じです。

5 **kadmin** コマンドを終了します。

```
kadmin: quit
```

例 25–16 サービス主体のキータブファイルへの追加

次の例では、`kadmin/kdc1.example.com` および `changepw/kdc1.example.com` 主体をマスター KDC のキータブファイルに追加しています。この例のキータブファイルは、`kdc.conf` ファイルで指定されている必要があります。

```
kdc1 # /usr/sbin/kadmin.local
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/kdc1.example.com changepw/kdc1.example.com
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type AES-256 CTS
          mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type AES-128 CTS
          mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: quit
```

次の例では、`denver` の `host` 主体を `denver` のキータブファイルに追加し、KDC が `denver` のネットワークサービスを認証できるようにします。

```
denver # /usr/sbin/kadmin
kadmin: ktadd host/denver.example.com
Entry for principal host/denver.example.com with kvno 3, encryption type AES-256 CTS
          mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
```

```
Entry for principal host/denver.example.com with kvno 3, encryption type Triple DES cbc mode
          with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

▼ キータブファイルからサービス主体を削除する方法

1. キータブファイルから削除するサービス主体が登録されているホスト上でスーパーユーザーになります。

2. **kadmin** コマンドを起動します。

   ```
   # /usr/sbin/kadmin
   ```

3. (省略可能) キータブファイル内の現在の主体 (鍵) の一覧を表示するには、**ktutil** コマンドを使用します。

   詳細な手順は、556 ページの「キータブファイル内のキー一覧 (主体) を表示する方法」を参照してください。

4. **ktremove** コマンドを使用して、キータブファイルから主体を削除します。

   ```
   kadmin: ktremove [-k keytab] [-q] principal [kvno | all | old ]
   ```

`-k` *keytab*	キータブファイルを指定します。デフォルトでは、`/etc/krb5/krb5.keytab` が使用されます。
`-q`	冗長な情報を表示しません。
principal	キータブファイルから削除する主体を指定します。
kvno	指定された主体のうち、鍵のバージョン番号が *kvno* と一致する主体のすべてのエントリを削除します。
`all`	指定された主体のすべてのエントリを削除します。
`old`	指定した主体のすべてのエントリを削除します。ただし、鍵のバージョン番号が最上位の主体は削除しません。

5. **kadmin** コマンドを終了します。

   ```
   kadmin: quit
   ```

例 25–17 キータブファイルからのサービス主体の削除

次の例では、`denver` の `host` 主体を `denver` のキータブファイルから削除します。

```
denver # /usr/sbin/kadmin
kadmin: ktremove host/denver.example.com@EXAMPLE.COM
kadmin: Entry for principal host/denver.example.com@EXAMPLE.COM with kvno 3
  removed from keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

▼ キータブファイル内のキー一覧 (主体) を表示する方法

1. キータブファイルが存在するホスト上でスーパーユーザーになります。

   注 – ほかのユーザーが所有するキータブファイルを作成することもできますが、キータブファイルのデフォルトの位置を使用するには root 所有権が必要です。

2. **ktutil** コマンドを起動します。

   ```
   # /usr/bin/ktutil
   ```

3. **read_kt** コマンドを使用して、キータブファイルをキー一覧バッファーに読み込みます。

   ```
   ktutil: read_kt keytab
   ```

4. **list** コマンドを使用して、キー一覧バッファーを表示します。

   ```
   ktutil: list
   ```

   現在のキー一覧バッファーが表示されます。

5. **ktutil** コマンドを終了します。

   ```
   ktutil: quit
   ```

例 25–18 キータブファイル内のキー一覧 (主体) の表示

次の例では、denver ホストの /etc/krb5/krb5.keytab ファイル内のキー一覧を表示します。

```
denver # /usr/bin/ktutil
    ktutil: read_kt /etc/krb5/krb5.keytab
    ktutil: list
slot KVNO Principal
---- ---- ---------------------------------------
   1    5 host/denver@EXAMPLE.COM
    ktutil: quit
```

▼ ホスト上のサービスの認証を一時的に無効にする方法

ネットワークアプリケーションサーバー上の `rlogin` や `ftp` など、サービスの認証メカニズムを一時的に無効にしなければならない場合があります。たとえば、保守作業中は、ユーザーがシステムにログインできないようにする必要があります。`ktutil` コマンドを使用してサーバーのキータブファイルからサービス主体を削除することにより、サービスの認証を一時的に無効にすることができます。このとき、`kadmin` 権限は必要ありません。認証を再度有効にするには、保存した元のキータブファイルを元の位置にコピーするだけです。

注 – デフォルトでは、ほとんどのサービスが認証を要求するように設定されています。そのように設定されていないときは、サービスの認証を無効にした場合でもサービスは動作します。

---

1 キータブファイルが存在するホスト上でスーパーユーザーになります。

注 – ほかのユーザーが所有するキータブファイルを作成することもできますが、キータブファイルのデフォルトの位置を使用するには `root` 所有権が必要です。

---

2 現在のキータブファイルを一時ファイルに保存します。

3 **`ktutil`** コマンドを起動します。

```
# /usr/bin/ktutil
```

4 **`read_kt`** コマンドを使用して、キータブファイルをキー一覧バッファーに読み込みます。

```
ktutil: read_kt keytab
```

5 **`list`** コマンドを使用して、キー一覧バッファーを表示します。

```
ktutil: list
```

現在のキー一覧バッファーが表示されます。無効にするサービスのスロット番号を記録します。

6 ホストのサービスを一時的に無効にするには、**`delete_entry`** コマンドを使用して、キー一覧バッファーから特定のサービス主体を削除します。

```
ktutil: delete_entry slot-number
```

この例では、*slot-number* に、削除するサービス主体のスロット番号を指定します。スロット番号は、`list` コマンドで表示できます。

7 **write_kt** コマンドを使用して、新しいキー一覧バッファーをキータブファイルに書き込みます。

```
ktutil: write_kt new-keytab
```

8 **ktutil** コマンドを終了します。

```
ktutil: quit
```

9 新しいキータブファイルの名前を変更します。

```
# mv new-keytab keytab
```

10 サービスを再度有効にする場合は、一時的な (元の) キータブファイルを元の場所にコピーします。

例 25–19 ホスト上のサービスを一時的に無効にする

次の例では、denver ホスト上の host サービスを一時的に無効にします。denver 上のホストサービスを再度有効にするには、 krb5.keytab.temp ファイルを /etc/krb5/krb5.keytab ファイルにコピーします。

```
denver # cp /etc/krb5/krb5.keytab /etc/krb5/krb5.keytab.temp
denver # /usr/bin/ktutil
    ktutil:read_kt /etc/krb5/krb5.keytab
    ktutil:list
slot KVNO Principal
---- ---- ---------------------------------------
   1    8 root/denver@EXAMPLE.COM
   2    5 host/denver@EXAMPLE.COM
    ktutil:delete_entry 2
    ktutil:list
slot KVNO Principal
---- ---- ---------------------------------------
   1    8 root/denver@EXAMPLE.COM
    ktutil:write_kt /etc/krb5/new.krb5.keytab
    ktutil: quit
denver # cp /etc/krb5/new.krb5.keytab /etc/krb5/krb5.keytab
```

◆ ◆ ◆ 第 26 章

Kerberos アプリケーションの使用 (手順)

この章は、Kerberos サービスが構成されているシステムのユーザーを対象としています。この章では、提供されている Kerberos コマンドとサービスの使用方法について説明します。この章を読み進めるには、これらのコマンド (Kerberos 以外) にすでに慣れ親しんでいる必要があります。

この章は一般的なユーザーを対象としているため、チケットの取得、表示、および廃棄について説明します。また、Kerberos パスワードの選択または変更についても説明します。

この章の内容は次のとおりです。

Oracle Solaris Kerberos 製品の概要については、第 21 章「Kerberos サービスについて」を参照してください。

Kerberos チケットの管理

この節では、チケットの取得、表示、および破棄を行う方法を説明します。チケットの概要については、402 ページの「Kerberos サービスの動作」を参照してください。

チケットを意識する必要があるか

すべての SEAM リリースまたは Oracle Solaris リリースがインストールされている場合、Kerberos は `login` コマンドに組み込まれており、チケットの取得はログイン時に自動的に行われます。Kerberos に対応するコマンドの `rsh`、 `rcp`、 `rdist`、 `telnet`、および `rlogin` は通常、チケットのコピーをほかのマシンに転送するように設定され

ています。そうすると、ユーザーがチケットを明示的に要求しなくても、それらのマシンにアクセスできるようになるからです。これがデフォルトの動作ですが、特定のユーザーの構成はこの自動転送を含んでいない場合があります。チケットの転送については、568 ページの「Kerberos コマンドの概要」および 571 ページの「Kerberos チケットの転送」を参照してください。

チケットの有効期限については、584 ページの「チケットの有効期限」を参照してください。

Kerberos チケットの作成

通常、PAM が適切に設定されている場合、ログインするとチケットが自動的に作成されるため、チケットを取得するために特別な作業をする必要はありません。ただし、チケットが期限切れになった場合は、チケットを作成する必要があります。また、デフォルトの主体のほかに別の主体を使用する必要がある場合 (たとえば、`rlogin -l` を使って他人としてマシンにログインする) があります。

チケットを作成するには、`kinit` コマンドを使用します。

```
% /usr/bin/kinit
```

`kinit` コマンドからはパスワードの入力を求めるプロンプトが表示されます。`kinit` コマンドの詳細な構文については、`kinit(1)` のマニュアルページを参照してください。

例 26–1 Kerberos チケットの作成

次の例では、ユーザー `jennifer` が自分のシステムにチケットを作成します。

```
% kinit
Password for jennifer@ENG.EXAMPLE.COM:       <Type password>
```

次の例では、ユーザー `david` が `-l` オプションを使用して 3 時間有効なチケットを作成します。

```
% kinit -l 3h david@EXAMPLE.ORG
Password for david@EXAMPLE.ORG:       <Type password>
```

次の例では、ユーザー `david` は、`-f` オプションを使用して、転送可能チケットを作成します。たとえば、この転送可能チケットを使用して、2 つ目のシステムにログインして、3 つ目のシステムに `telnet` を実行できます。

```
% kinit -f david@EXAMPLE.ORG
Password for david@EXAMPLE.ORG:          <Type password>
```

例 26–1 Kerberos チケットの作成 (続き)

転送チケットをどのように使用するかについては、571 ページの「Kerberos チケットの転送」および 582 ページの「チケットの種類」を参照してください。

Kerberos チケットの表示

すべてのチケットが同じ属性を持つわけではありません。チケットの属性には、「転送可能 (Forwardable)」、「遅延 (Postdated)」などがあります。また、1 つのチケットに「転送可能」と「遅延」の両方が指定されていることもあります。現在のチケットが何で、どのような属性を持つかを知るには、`klist` コマンドで `-f` オプションを使用します。

```
% /usr/bin/klist -f
```

次の記号はチケットに関連付けられる属性です。`klist` によって表示されます。

A 事前認証済み

D 遅延可能 (Postdatable)

d 遅延 (Postdated)

F 転送可能 (Forwardable)

f 転送済み (Forwarded)

I 初期 (Initial)

i 無効 (Invalid)

P プロキシ可能 (Proxiable)

p プロキシ (Proxy)

R 更新可能 (Renewable)

チケットに指定できる属性については、582 ページの「チケットの種類」を参照してください。

例 26–2 Kerberos チケットの表示

次の例は、ユーザー `jennifer` の初期チケットが転送可能 (F) と遅延 (d) のプロパティーを持っていて、まだ検証されていないこと (i) を示します。

```
% /usr/bin/klist -f
Ticket cache: /tmp/krb5cc_74287
Default principal: jennifer@EXAMPLE.COM

Valid starting                Expires                Service principal
```

例 26–2 Kerberosチケットの表示 *(続き)*

```
09 Mar 04 15:09:51  09 Mar 04 21:09:51  nfs/EXAMPLE.COM@EXAMPLE.COM
        renew until 10 Mar 04 15:12:51, Flags: Fdi
```

次の例は、ユーザー david が別のホストから自分のホストに転送済み (f) チケットを 2 つ持っていることを示します。これらのチケットは転送可能 (F) です。

```
% klist -f
Ticket cache: /tmp/krb5cc_74287
Default principal: david@EXAMPLE.COM

Valid starting                 Expires                 Service principal
07 Mar 04 06:09:51  09 Mar 04 23:33:51  host/EXAMPLE.COM@EXAMPLE.COM
        renew until 10 Mar 04 17:09:51, Flags: fF

Valid starting                 Expires                 Service principal
08 Mar 04 08:09:51  09 Mar 04 12:54:51  nfs/EXAMPLE.COM@EXAMPLE.COM
        renew until 10 Mar 04 15:22:51, Flags: fF
```

次の例は、-e オプションを使用してセッション鍵の暗号化タイプとチケットを表示させる方法を示しています。-a オプションを使用すると、ネームサービスが変換を行える場合、ホストアドレスをホスト名に変換できます。

```
% klist -fea
Ticket cache: /tmp/krb5cc_74287
Default principal: david@EXAMPLE.COM

Valid starting                 Expires                 Service principal
07 Mar 04 06:09:51  09 Mar 04 23:33:51  krbtgt/EXAMPLE.COM@EXAMPLE.COM
        renew until 10 Mar 04 17:09:51, Flags: FRIA
        Etype(skey, tkt): DES cbc mode with RSA-MD5, DES cbc mode with CRC-32
        Addresses: client.example.com
```

Kerberosチケットの破棄

現在のセッション中に取得したすべての Kerberos チケットを破棄するには、kdestroy コマンドを使用します。このコマンドは、資格キャッシュを破棄して、すべての資格とチケットを破棄します。通常、これは必要ありませんが、kdestroy コマンドを実行すると、ログインしていないときに資格キャッシュが継続して存在する可能性を減らすことができます。

チケットを破棄するには、kdestroy コマンドを使用します。

```
% /usr/bin/kdestroy
```

kdestroy コマンドは、そのユーザーのすべてのチケットを破棄します。このコマンドを使用して、特定のチケットを選択して破棄することはできません。

システムを離れるときに侵入者が権限を使用する危険がある場合は、kdestroy を使用してチケットを破棄するか、スクリーンセーバーを使って画面をロックする必要があります。

Kerberos パスワード管理

Kerberos サービスが構成されている場合、パスワードを 2 つ持つことになります。通常の Solaris パスワードと Kerberos パスワードです。これらのパスワードは同じでも、異なっていても構いません。

パスワード選択のヒント

パスワードには、キーボードから入力できるほとんどの文字を使用できます。ただし、Ctrl キーと Return キーは使用できません。良いパスワードとは、覚えやすく、しかも他人が簡単に推定できないパスワードです。悪いパスワードの例を次に示します。

- 辞書に出てくる言葉
- よく見られるありふれた名前
- 有名な人やキャラクタの名前
- ユーザーの氏名またはユーザー名 (たとえば、名前を逆に綴る、2 度繰り返す)
- 配偶者の名前、子供の名前、ペットの名前
- 自分の誕生日や親戚の誕生日
- 社会保険番号、運転免許証番号、パスポート番号、またはこれに類した身分証明書番号
- このマニュアルやほかのマニュアルに出てくるサンプルパスワード

良いパスワードとは 8 文字以上の長さで、大文字、小文字、数字、句読記号などが混在しているものです。次に例を示します。

- 「I2LMHinSF」などの短縮形。(「I too left my heart in San Francisco」と覚える)
- 「WumpaBun」、「WangDangdoodle!」など、発音しやすい意味のない語句
- 「6o'cluck」、「RrriotGrrrlsRrrule!」など、わざとスペルを間違えた語句

注意 – これらの例は使用しないでください。マニュアルの例に使用されているパスワードは侵入者が最初に試みるパスワードです。

パスワードの変更方法

PAM が適切に設定されている場合、Kerberos パスワードは次の 2 つの方法で変更できます。

- 通常の UNIX の passwd コマンド。Kerberos サービスが構成されていると、passwd コマンドでも新しい Kerberos パスワードを求めるプロンプトが自動的に表示されます。

  kpasswd の代わりに passwd を使用する利点は、UNIX と Kerbeors 両方のパスワードを同時に設定できることです。ただし、一般的には passwd コマンドで両方のパスワードを同時に変更する必要はありません。UNIX パスワードだけを変更して Kerberos パスワードは変更しなかったり、その逆であっても構いません。

  注 – passwd コマンドの動作は、PAM モジュールの構成方法によって異なります。構成によっては、両方のパスワードを変更しなければならない場合があります。あるサイトでは UNIX パスワードの変更が必須であり、別のサイトでは Kerberos パスワードの変更が必須であるという場合があります。

- kpasswd コマンド。kpasswd コマンドは、passwd コマンドと似ています。ただし、kpasswd コマンドでは、Kerberos パスワード以外は変更できません。UNIX パスワードを変更する場合は、passwd コマンドを使用する必要があります。

  もう 1 つの違いは、kpasswd コマンドでは、有効な UNIX ユーザーではない Kerberos 主体のパスワードを変更できる点です。たとえば、david/admin は Kerberos 主体ですが、実際の UNIX ユーザーではありません。したがって、この場合は、passwd コマンドの代わりに kpasswd コマンドを使用する必要があります。

パスワードを変更しても、変更がシステム全体に伝播されるまでには、ある程度の時間が必要です (特に大規模なネットワークでは)。システムの設定方法によりますが、この時間は数分から 1 時間以上になることがあります。パスワードを変更したあとすぐに新しい Kerberos チケットを取得する場合は、新しいパスワードをまず試してください。新しいパスワードが有効でない場合は、以前のパスワードを使用して再度試してください。

Kerberos V5 プロトコルでは、システム管理者が有効なパスワードの基準をユーザーごとに設定できます。この基準は、ユーザーごとの「ポリシー」に定義できます。デフォルトのポリシーを使用することもできます。ポリシーの詳細については、539 ページの「Kerberos 主体の管理」を参照してください。

たとえば、ユーザー jennifer の jenpol ポリシーでは、パスワードは 8 文字以上で、2 種類以上の文字で構成されると定義されているとします。その場合、パスワードとして「sloth」を入力すると、kpasswd によって拒否されます。

```
% kpasswd
kpasswd: Changing password for jennifer@ENG.EXAMPLE.COM.
Old password:        <Jennifer types her existing password>
kpasswd: jennifer@ENG.EXAMPLE.COM's password is controlled by
the policy jenpol
which requires a minimum of 8 characters from at least 2 classes
(the five classes are lowercase, uppercase, numbers, punctuation,
and all other characters).
New password:      <Jennifer types 'sloth'>
New password (again):       <Jennifer re-types 'sloth'>
kpasswd: New password is too short.
Please choose a password which is at least 4 characters long.
```

次に、jennifer はパスワードとして「slothrop49」を入力します。「slothrop49」は長さが 8 文字以上で、2 種類の文字 (数字と小文字) が混在しているため基準に合っています。

```
% kpasswd
kpasswd: Changing password for jennifer@ENG.EXAMPLE.COM.
Old password:       <Jennifer types her existing password>
kpasswd: jennifer@ENG.EXAMPLE.COM's password is controlled by
the policy jenpol
which requires a minimum of 8 characters from at least 2 classes
(the five classes are lowercase, uppercase, numbers, punctuation,
and all other characters).
New password:       <Jennifer types 'slothrop49'>
New password (again):       <Jennifer re-types 'slothrop49'>
Kerberos password changed.
```

例 **26–3** パスワードの変更方法

次の例では、ユーザー david が passwd を使用して、UNIX および Kerberos のパスワードを変更します。

```
% passwd
    passwd:  Changing password for david
    Enter login (NIS+) password:                <Type the current UNIX password>
    New password:                           <Type the new UNIX password>
    Re-enter password:                      <Confirm the new UNIX password>
    Old KRB5 password:                      <Type the current Kerberos password>
    New KRB5 password:                      <Type the new Kerberos password>
    Re-enter new KRB5 password:             <Confirm the new Kerberos password>
```

passwd では、UNIX パスワードと Kerberos パスワードの両方が要求されることに注意してください。この動作は、デフォルトの設定です。この場合、ユーザー david は kpasswd を使用して、次の例で示すように Kerberos パスワードを変更する必要があります。

例 26–3 パスワードの変更方法 (続き)

次の例では、ユーザー david が kpasswd を使用して、Kerberos パスワードだけを変更します。

```
% kpasswd
kpasswd: Changing password for david@ENG.EXAMPLE.COM.
Old password:                <Type the current Kerberos password>
New password:                <Type the new Kerberos password>
New password (again):        <Confirm the new Kerberos password>
Kerberos password changed.
```

次の例では、ユーザー david が、Kerberos 主体 david/admin (有効な UNIX ユーザーではない) のパスワードを変更します。kpasswd を使用する必要があります。

```
% kpasswd david/admin
kpasswd:  Changing password for david/admin.
Old password:                <Type the current Kerberos password>
New password:                <Type the new Kerberos password>
New password (again):        <Type the new Kerberos password>
Kerberos password changed.
```

アカウントへのアクセス認可

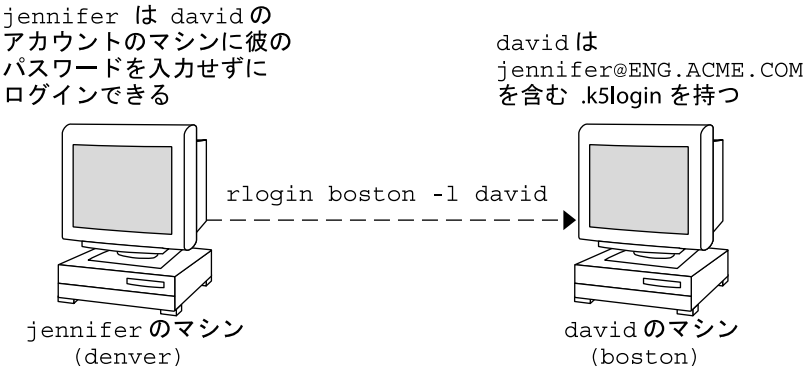
他人があなたのアカウントを使用して (あなたとして) ログインする必要がある場合、Kerberos を使用すれば、パスワードを公表せずにそれを実現できます。それには、ホームディレクトリに .k5login ファイルを格納します。.k5login ファイルは、アクセス認可を与える各ユーザーに対応する、1 つ以上の Kerberos 主体の一覧です(1 つの主体を 1 行に入力する必要があります)。

ユーザー david が次のような .k5login ファイルをホームディレクトリに格納しているとします。

```
jennifer@ENG.EXAMPLE.COM
joe@EXAMPLE.ORG
```

このファイルによって、ユーザー jennifer と joe は david として振る舞うことができます。ただしそれには、その二人がすでにそれぞれのレルムにおいて Kerberos チケットを取得している必要があります。たとえば、jennifer は david として、david のマシン (boston) に david 自身のパスワードを入力せずに遠隔ログインできます。

図 26–1　アカウントへのアクセスを認可する .k5login ファイルの使用



david のホームディレクトリが、Kerberos V5 プロトコル経由で別の (3 つ目の) マシンから NFS マウントされている場合、jennifer がそのホームディレクトリにアクセスするには、彼女が転送チケットを所持している必要があります。転送可能チケットの使用例については、560 ページの「Kerberos チケットの作成」を参照してください。

ネットワーク上のほかのマシンにログインする必要がある場合、それらのマシン上の .k5login ファイル内に自分の Kerberos 主体を追加します。

.k5login ファイルの使用は、次の理由により、パスワードを公表するよりも安全です。

- .k5login ファイルから特定の主体を削除することで、特定ユーザーのアクセス認可をいつでも無効にできます。
- ホームディレクトリ内の .k5login ファイルにある名前の付いたユーザーの主体は、そのマシン (NFS 上などで .k5login ファイルを共有している場合には一連のマシン) 上のアカウントへの完全なアクセス権を持ちます。ただし、Kerberos サービスは、そのユーザー ID に基づいてアクセスを承認します。つまり、jennifer は、joe のマシンにログインして、そこで作業を行えます。ただし、ftp や rlogin などの Kerberos プログラムを使用する場合は、自分自身のアカウントで使用します。
- Kerberos は、チケットを取得したユーザーのログを保持しているため、システム管理者は、特定の時刻に特定のユーザー ID を使用できるユーザーを、必要に応じて調べることができます。

.k5login ファイルの一般的な使い方の 1 つは、このファイルを root のホームディレクトリに格納し、ファイル内に記述された Kerberos 主体にそのマシンの root アクセス権限を与える、というものです。この設定により、システム管理者

は、スーパーユーザーのパスワードを公表したり、そのパスワードをネットワーク上で入力したりせずに、ローカルで root になることも、root として遠隔ログインすることもできるようになります。

例 26–4 アカウントへのアクセスを認可する .k5login ファイルの使用

jennifer が、マシン boston.example.com に root としてログインするとします。彼女の主体名は boston.example.com 上の root ホームディレクトリ内の .k5login ファイルに含まれているため、彼女はここでもパスワードを入力する必要がありません。

```
% rlogin boston.example.com -l root -x
This rlogin session is using DES encryption for all data transmissions.
Last login: Thu Jun 20 16:20:50 from daffodil
SunOS Release 5.7 (GENERIC) #2: Tue Nov 14 18:09:31 EST 1998
boston[root]%
```

Kerberos ユーザーコマンド

Kerberos V5 製品は「シングルサインオン」システムです。つまり、1 度だけパスワードを入力する必要があるという意味です。Kerberos V5 プログラムがユーザーに代わって認証 (オプションで暗号化も) 行います。これは、Kerberos が、既存のよく知られたネットワークプログラム群ごとに構築されているためです。Kerberos V5 アプリケーションは、既存の UNIX ネットワークプログラムに Kerberos 機能を追加したバージョンです。

たとえば、ある Kerberos プログラムを使って特定の遠隔ホストに接続する場合、そのプログラム、KDC、 その遠隔ホストの 3 者は、一連のネゴシエーションをすばやく実行します。これらのネゴシエーションが正常終了した場合、そのプログラムは遠隔ホストに対し、そのユーザーの身元をユーザーに代わって証明したことになり、遠隔ホストはそのユーザーにアクセスを認可します。

Kerberos コマンドは最初に、Kerberos を使用して認証しようとする点に注意してください。Kerberos 認証が失敗すると、エラーが発生するか、UNIX 認証が試みられますが、どちらになるかは、コマンドに指定されるオプションによって決まります。詳細は、各 Kerberos コマンドのマニュアルページの「Kerberos Security」節を参照してください。

Kerberos コマンドの概要

Kerberos ネットワークサービスは、インターネット上のほかのマシンと接続するプログラムです。これらには、次のプログラムがあります。

- ftp
- rcp

- rdist
- rlogin
- rsh
- ssh
- telnet

これらのプログラムは、Kerberos チケットを透過的に使って認証 (と暗号化) のネゴシエーションを遠隔ホストと行うための機能も備えています。それらを使用する際の変化といえば、多くの場合、パスワードを入力する必要がなくなったことに気付くことぐらいです。このとき、Kerberos がユーザーに代わってユーザーの身元証明を行なってくれています。

Kerberos V5 ネットワークプログラムは、次の機能を実現するオプションを備えています。

- 別のホストへチケットを転送します (最初に転送可能チケットを取得していた場合)。
- ユーザーと遠隔ホスト間で送受信されるデータを暗号化します。

注 – この節では、読者がすでにこれらの Kerberos 以外のプログラムに慣れ親しんでいることを前提に、Kerberos V5 パッケージによって追加された Kerberos 機能だけに的を絞って説明しています。ここで説明するコマンドの詳細については、各コマンドのマニュアルページを参照してください。

次の Kerberos オプションが、ftp、 rcp、 rlogin、 rsh、 および telnet に追加されています。

-a　既存のチケットを使用して自動ログインしようとします。getlogin() によって返されたユーザー名を使用します。ただし、現在のユーザー ID と異なる場合には使用しません。詳細は、telnet(1) のマニュアルページを参照してください。

-f　「再転送不可能な」チケットを遠隔ホストに転送します。このオプションは、-F オプションと互いに排他の関係にあります。つまり、これらを同一コマンド内で同時に使用することはできません。

3 つ目のホスト上のほかの Kerberos に基づくサービスに自身を認証する必要がある場合、チケットを転送します。たとえば、ほかのマシンに遠隔ログインして、そこから 3 つ目のマシンに遠隔ログインします。

遠隔ホスト上のホームディレクトリが Kerberos V5 を使用して NFS マウントされている場合、転送可能チケットを使用する必要があります。そうしなかった場合、ホームディレクトリにアクセスできません。つまり、最初、システム 1 にログインするものとします。シ

ステム 3 からホームディレクトリをマウントしたホームマシン、システム 2 にシステム 1 から遠隔ログインします。`rlogin` で `-f` または `-F` オプションを使用しない場合、チケットがシステム 3 に転送されないため、ホームディレクトリを取得できません。

`kinit` はデフォルトで、転送可能なチケット認可チケット (TGT) を取得します。ただし、このような構成を変更することは可能です。

チケットの転送の詳細は、571 ページの「Kerberos チケットの転送」を参照してください。

`-F` 「再転送可能な」TGT のコピーを遠隔システムに転送します。このオプションは、`-f` と似ていますが、さらに先のマシン (たとえば 4 つ目や 5 つ目のマシン) へのアクセスを可能とする点が異なります。したがって、`-F` オプションは `-f` オプションのスーパーセットであると考えられます。`-F` オプションと `-f` オプションは、互いに排他の関係にあります。つまり、これらを同一コマンド内で同時に使用することはできません。

チケットの転送の詳細は、571 ページの「Kerberos チケットの転送」を参照してください。

`-k` *realm* `krb5.conf` ファイルを使用してレルム自身を決定する代わりに、特定の *realm* 内の遠隔ホストのチケットを要求します。

`-K` チケットを使用して遠隔ホストへの認証を行いますが、自動ログインは行いません。

`-m` *mechanism* `/etc/gss/mech` ファイルの記載どおりに、使用する GSS-API セキュリティーメカニズムを指定します。デフォルトは `kerberos_v5`。

`-x` このセッションを暗号化します。

`-X` *auth-type* *auth-type* タイプの認証を無効にします。

次の表は、コマンド固有のオプションを示します。「X」は、コマンドがそのオプションを持つことを示します。

表 **26–1** ネットワークコマンドの Kerberos オプション

	ftp	rcp	rlogin	rsh	telnet
`-a`					X
`-f`	X		X	X	X
`-F`			X	X	X

表 26–1 ネットワークコマンドの Kerberos オプション (続き)

	ftp	rcp	rlogin	rsh	telnet
-k		X	X	X	X
-K					X
-m	X				
-x	X	X	X	X	X
-X					X

さらに、ftp では、ftp のプロンプト対してコマンドを入力することでにセッションの保護レベルを設定できます。

clear 保護レベルを「clear」(保護なし) に設定します。この保護レベルがデフォルトです。

private 保護レベルを「private」に設定します。データ転送は、暗号化により機密性と完全性が保護されます。ただし、この機密サービスは、Kerberos ユーザーによっては利用できない可能性もあります。

safe 保護レベルを「safe」に設定します。データ転送は、暗号チェックサムによって完全性が保護されます。

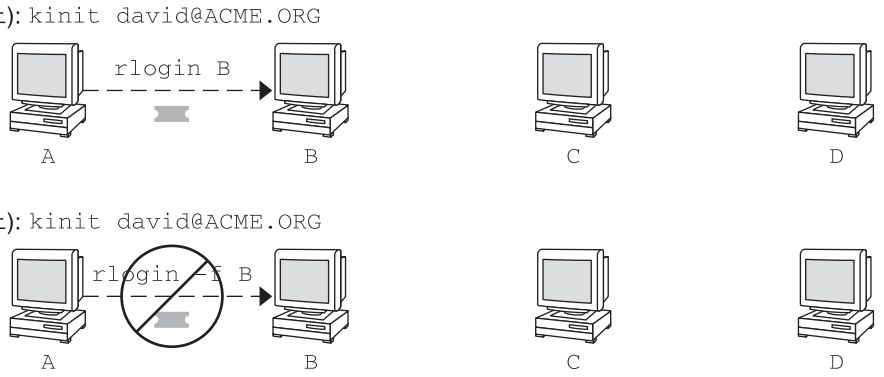
ftp プロンプトで protect と入力したあとに上記に示した保護レベル (clear、private、 または safe) のいずれかを入力して、保護レベルを設定することもできます。

Kerberos チケットの転送

568 ページの「Kerberos コマンドの概要」 で説明したように、一部のコマンドで -f または -F オプションを使用するとチケットを転送できます。チケットを転送すると、ネットワークトランザクションの「チェーン化」が可能となります。たとえば、あるマシンに遠隔ログインして、そこから別のマシンに遠隔ログインできます。-f オプションではチケットを転送することができますが、-F オプションでは転送済みのチケットを再転送することが可能です。

図 26–2 で、ユーザー david は kinit を使用して、転送不可能なチケット認可チケット (TGT) を取得します。-f オプションを指定しなかったため、チケットは転送不可能なチケットです。シナリオ 1 では、マシン B に遠隔ログインできますが、それ以上遠隔ログインできません。シナリオ 2 の rlogin -f コマンドは失敗します。なぜなら、彼は転送不可能なチケットを転送しようとしているからです。

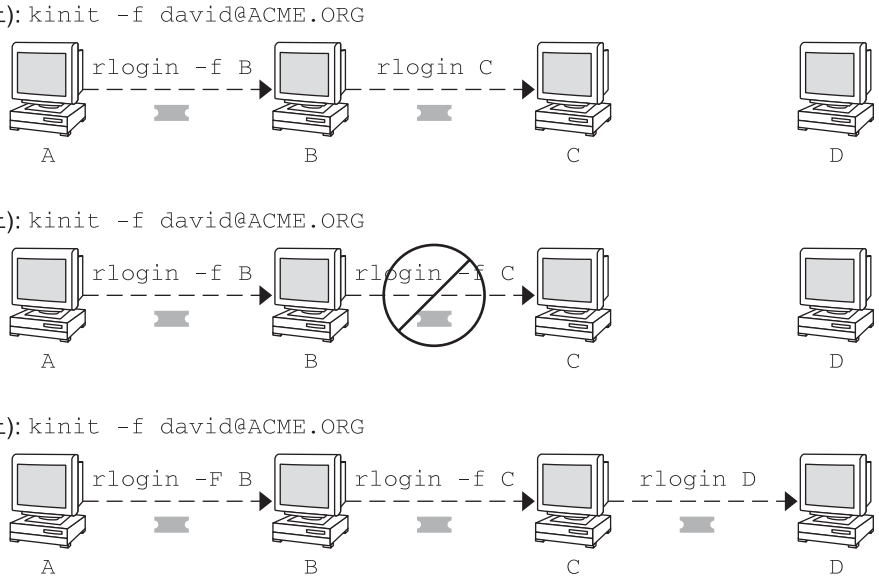
図 26–2 転送不可能なチケットの使用



実際にはデフォルトで、kinit が転送可能なチケットを取得するように Kerberos 構成ファイルが設定されます。ただし、ユーザーの構成はこれと異なっても構いません。説明するために、TGT が kinit -f で起動されない限り、kinit は転送可能な TGT を取得できないと想定します。kinit には、-F オプションがないことに注意してください。TGT は、転送可能、転送不可能のいずれかです。

図 26–3 で、ユーザー david は、kinit -f を使用して、転送可能な TGT を取得します。シナリオ 3 では、rlogin で転送可能なチケットを使用するため、マシン C に到達できます。シナリオ 4 の 2 回目の rlogin は失敗します。なぜなら、このチケットは再転送可能ではないからです。シナリオ 5 のように -F オプションを代わりに使用すれば、2 回目の rlogin は成功し、チケットがマシン D に再転送されます。

図 26–3 転送可能チケットの使用



Kerberos コマンドの使用 (例)

次の例は、Kerberos コマンドのオプションの使用方法を示しています。

例 26–5 telnet で -a、 -f、 および -x オプションを使用する

この例では、ユーザー david はすでにログインしており、マシン denver.example.com に telnet しようとしています。彼は、-f オプションを使って既存のチケットを転送し、-x オプションを使ってセッションを暗号化し、-a オプションを使って自動ログインを実行します。3 つ目のホストのサービスを使用する予定はないため、ここでは -F ではなく -f を使用しています。

```
% telnet -a -f -x denver.example.com
Trying 128.0.0.5...
Connected to denver.example.com. Escape character is '^]'.
[ Kerberos V5 accepts you as "david@eng.example.com" ]
[ Kerberos V5 accepted forwarded credentials ]
SunOS 5.9: Tue May 21 00:31:42 EDT 2004  Welcome to SunOS
%
```

david のマシンが、Kerberos を使用して denver.example.com に対する彼の認証を行い、さらに自動ログインしている点に注意してください。彼は、暗号化されたセッションと、いつでも利用可能なチケットのコピーとを手に入れましたが、パスワードを入力する必要は一度もありませんでした。Kerberos 以外のバージョンの telnet を使用している場合、パスワードの入力を要求され、そのパスワードは暗号

例 26–5 telnet で -a、 -f、 および -x オプションを使用する (続き)

化されていないネットワーク上で送信されます。その時点で侵入者がネットワークトラフィックを監視していると、侵入者は david のパスワードを知ることになります。

Kerberos チケットを転送した場合、telnet (およびここで説明したその他のコマンド) は、終了時にそのチケットを破棄します。

例 26–6 rlogin で -F オプションを使用する

ここでは、ユーザー jennifer が自身のマシン boston.example.com にログインするものとします。-F オプションを使用して既存のチケットを転送し、-x オプションを使用してセッションを暗号化します。-f ではなく -F を選択します。これは boston にログインしたあとで、再転送されるチケットが必要な別のネットワークトランザクションを実行するためです。また、既存のチケットを転送するので、パスワードを入力する必要はありません。

```
% rlogin boston.example.com -F -x
This rlogin session is using encryption for all transmissions.
Last login Mon May 19 15:19:49 from daffodil
SunOS Release 5.9 (GENERIC) #2 Tue Nov 14 18:09:3 EST 2003
%
```

例 26–7 ftp で保護レベルを設定する

joe が ftp を使用して、マシン denver.example.com のディレクトリ ~joe/MAIL から自分宛てのメールを取得します。ただし、セッションを暗号化します。やり取りは次のようになります。

```
% ftp -f denver.example.com
Connected to denver.example.com
220 denver.example.org FTP server (Version 6.0) ready.
334 Using authentication type GSSAPI; ADAT must follow
GSSAPI accepted as authentication type
GSSAPI authentication succeeded Name (daffodil.example.org:joe)
232 GSSAPI user joe@MELPOMENE.EXAMPLE.COM is authorized as joe
230 User joe logged in.
Remote system type is UNIX.
Using BINARY mode to transfer files.
ftp> protect private
200 Protection level set to Private
ftp> cd ~joe/MAIL
250 CWD command successful.
ftp> get RMAIL
227 Entering Passive Mode (128,0,0,5,16,49)
150 Opening BINARY mode data connection for RMAIL (158336 bytes).
226 Transfer complete. 158336 bytes received in 1.9 seconds (1.4e+02 Kbytes/s)
ftp> quit
%
```

例 26–7　ftp で保護レベルを設定する　(続き)

セッションを暗号化するために、joe は保護レベルを private に設定しています。

◆ ◆ ◆ 第 27 章

Kerberos サービス (参照)

この章では、Kerberos 製品に組み込まれている多数のファイル、コマンド、およびデーモンを示します。また、Kerberos 認証の機能についても詳細に説明します。

この章の内容は次のとおりです。

Kerberos ファイル

表 27–1 Kerberos ファイル

ファイル名	説明
~/.gkadmin	SEAM ツールで新しい主体を作成するときのデフォルト値
~/.k5login	Kerberos アカウントに対してアクセス権を許可する主体のリスト
/etc/krb5/kadm5.acl	Kerberos アクセス制御リストファイル。KDC 管理者の主体名とその Kerberos 管理特権を含みます
/etc/krb5/kadm5.keytab	マスター KDC 上の kadmin サービスのキータブファイル

表 27–1 Kerberos ファイル (続き)

ファイル名	説明
/etc/krb5/kdc.conf	KDC 構成ファイル
/etc/krb5/kpropd.acl	Kerberos データベース伝播構成ファイル
/etc/krb5/krb5.conf	Kerberos レルム構成ファイル
/etc/krb5/krb5.keytab	ネットワークアプリケーションサーバー用キータブファイル
/etc/krb5/warn.conf	Kerberos チケットの有効期限切れの警告と自動更新の構成ファイル
/etc/pam.conf	PAM 構成ファイル
/tmp/krb5cc_*uid*	デフォルト資格キャッシュ (*uid* はユーザーの 10 進 UID)
/tmp/ovsec_adm.*xxxxxx*	パスワード変更操作の間だけ有効な一時資格キャッシュ (*xxxxxx* はランダムな文字列)
/var/krb5/.k5.*REALM*	KDC stash ファイル。KDC マスター鍵のコピーを含みます
/var/krb5/kadmin.log	kadmind のログファイル
/var/krb5/kdc.log	KDC のログファイル
/var/krb5/principal	Kerberos 主体データベース
/var/krb5/principal.kadm5	Kerberos 管理データベース。ポリシー情報を含みます
/var/krb5/principal.kadm5.lock	Kerberos 管理データベースのロックファイル
/var/krb5/principal.ok	Kerberos 主体データベースの初期化ファイル。Kerberos データベースの初期化が正常終了すると作成されます
/var/krb5/principal.ulog	Kerberos 更新ログ。増分伝播の更新を含みます
/var/krb5/slave_datatrans	KDC のバックアップファイルで、kprop_script スクリプトが伝播時に使用します
/var/krb5/slave_datatrans_*slave*	*slave* に完全更新が行われるときに作成される一時ダンプファイル

Kerberos コマンド

この節では、Kerberos 製品に含まれているコマンドの一部を示します。

表 27–2 Kerberos コマンド

コマンド	説明
/usr/bin/ftp	ファイル転送プロトコルプログラム
/usr/bin/kdestroy	Kerberos チケットを破棄します
/usr/bin/kinit	Kerberos チケット認可チケットを取得し、キャッシュに格納します
/usr/bin/klist	現在の Kerberos チケットを表示します
/usr/bin/kpasswd	Kerberos パスワードを変更します
/usr/bin/ktutil	Kerberos キータブファイルを管理します
/usr/bin/rcp	遠隔ファイルコピープログラム
/usr/bin/rdist	遠隔ファイル配布プログラム
/usr/bin/rlogin	遠隔ログインプログラム
/usr/bin/rsh	遠隔シェルプログラム
/usr/bin/telnet	Kerberos telnet プログラム
/usr/lib/krb5/kprop	Kerberos データベース伝播プログラム
/usr/sbin/gkadmin	Kerberos データベース管理 GUI プログラム。主体とポリシーの管理に使用されます
/usr/sbin/gsscred	gsscred テーブルエントリを管理します
/usr/sbin/kadmin	遠隔 Kerberos データベース管理プログラム (Kerberos 認証とともに実行)。主体、ポリシー、およびキータブファイルの管理に使用されます
/usr/sbin/kadmin.local	ローカル Kerberos データベース管理プログラム (Kerberos 認証なしで動作するが、マスター KDC 上で実行する必要がある)。主体、ポリシー、およびキータブファイルの管理に使用されます
/usr/sbin/kclient	Kerberos クライアントのインストールスクリプト。インストールプロファイルとともに、またはなしで使用されます
/usr/sbin/kdb5_ldap_util	Kerberos データベースの LDAP コンテナを作成します
/usr/sbin/kdb5_util	Kerberos データベースと stash ファイルを作成します
/usr/sbin/kgcmgr	Kerberos のマスター KDC とスレーブ KDC を構成します

表 27–2 Kerberos コマンド (続き)

コマンド	説明
/usr/sbin/kproplog	更新ログ内の更新エントリの概要を一覧表示します

Kerberos デーモン

次の表は、Kerberos 製品で使用されるデーモンの一覧です。

表 27–3 Kerberos デーモン

デーモン	説明
/usr/sbin/in.ftpd	ファイル転送プロトコルデーモン
/usr/lib/krb5/kadmind	Kerberos データベース管理デーモン
/usr/lib/krb5/kpropd	Kerberos データベース伝播デーモン
/usr/lib/krb5/krb5kdc	Kerberos チケット処理デーモン
/usr/lib/krb5/ktkt_warnd	Kerberos チケットの有効期限切れの警告と自動更新のデーモン
/usr/sbin/in.rlogind	遠隔ログインデーモン
/usr/sbin/in.rshd	遠隔シェルデーモン
/usr/sbin/in.telnetd	telnet デーモン

Kerberos の用語

この節では、Kerberos の用語とその定義について説明します。これらの用語は、Kerberos のマニュアル全体で使用されます。Kerberos の概念を理解するには、これらの用語を理解する必要があります。

Kerberos 固有の用語

KDC を管理するには、この節で説明する用語を理解する必要があります。

「鍵配布センター (*Key Distribution Center*、 *KDC*)」は、資格の発行を担当する Kerberos の構成要素です。資格は、KDC データベースに格納されている情報に基づいて作成されます。各レルムには 2 つ以上の KDC サーバー (マスターと 1 つ以上のスレーブ) が必要です。すべての KDC が資格を生成できますが、KDC データベースを変更できるのはマスター KDC だけです。

KDCのマスター鍵は*stash*ファイルに含まれます。サーバーがリブートされると、この鍵を使用してKDCが自動的に認証されて、`kadmind`と`krb5kdc`コマンドが起動されます。このファイルにはマスター鍵が入っているため、このファイルやバックアップは安全な場所に保管する必要があります。ファイルは、`root`の読み取り専用のアクセス権で作成されます。ファイルをセキュリティー保護するには、アクセス権を変更しないでください。ファイルの保護が破られると、この鍵を使用してKDCデータベースのアクセスや変更が可能になります。

認証固有の用語

認証処理を理解するには、この節で説明する用語を理解する必要があります。プログラマやシステム管理者はこれらの用語に精通している必要があります。

「クライアント」は、ユーザーのワークステーションで動作するソフトウェアです。クライアントで動作するKerberosソフトウェアは、処理中に多数の要求を作成します。このため、Kerberosソフトウェアとユーザーの動作を区別することが重要です。

「サーバー」と「サービス」はよく同じ意味で使われます。明確に定義すると、「サーバー」は、Kerberosソフトウェアが動作する物理システムです。「サービス」とは、サーバー上でサポートされる特定の機能(`ftp`や`nfs`)です。サーバーがサービスの一部として記述されることがよくありますが、これはこれらの用語の定義をあいまいにします。このドキュメントでは、サーバーという用語は、物理システムを指します。サービスという用語は、ソフトウェアを指します。

Kerberos製品は、2種類の鍵を使用します。1つはパスワード由来の鍵です。パスワード由来の鍵は、各ユーザー主体に与えられ、そのユーザーとKDCだけに知られています。Kerberos製品が使用するもう1つの鍵は、パスワードとの関連性がないランダム鍵です。そのため、ユーザー主体による使用には適しません。通常、ランダム鍵は、キータブにエントリがあり、KDCによって作成されるセッション鍵を持つサービス主体で使用されます。サービスは非対話形式での実行を許可するキータブファイル内の鍵にアクセスできるため、サービス主体はランダム鍵を使用できます。セッション鍵は、クライアントとサービス間のトランザクションを保護するためにKDCによって生成され、クライアントとサービス間で共有されます。

「チケット」は、ユーザーの識別情報をサーバーやサービスに安全に渡すために使用される情報パケットです。チケットは、単一クライアントと特定サーバー上の特定サービスだけに有効です。チケットには、次のものが含まれます。

- サービスの主体名
- ユーザーの主体名
- ユーザーのホストのIPアドレス
- タイムスタンプ
- チケットの有効期限を定義する値

- セッション鍵のコピー

これらのすべてのデータは、サーバーのサービス鍵に暗号化されます。KDC は、次に説明する資格に組み込まれたチケットを発行します。チケットは、発行されてから有効期限まで再使用できます。

「資格」は、チケットとそれに対応するセッション鍵を含む情報パケットです。資格は要求する主体の鍵で暗号化されます。一般的に、KDC はクライアントからのチケット要求に応じて資格を生成します。

「オーセンティケータ *(authenticator)*」は、クライアントのユーザー主体を認証するためにサーバーが使用する情報です。オーセンティケータは、ユーザーの主体名、タイムスタンプ、およびその他のデータを含みます。チケットとは異なり、オーセンティケータは一度しか使用できません。通常、サービスへのアクセスが要求されたときに使用されます。オーセンティケータは、クライアントとサーバーが共有するセッション鍵を使用して暗号化されます。通常、クライアントが、オーセンティケータを作成し、サーバーまたはサービスに対して認証するためにサーバーまたはサービスのチケットとともに送信します。

チケットの種類

チケットには、チケットがどのように使用されるかを決めるプロパティーがあります。これらのプロパティーは、チケットの作成時にチケットに割り当てられます。ただし、チケットのプロパティーはあとから変更できます。たとえば、チケットは、転送可能から転送済みに変更できます。チケットのプロパティーを表示するには、`klist` コマンドを使用します561 ページの「Kerberos チケットの表示」を参照してください。

チケットは、次の 1 つまたは複数のプロパティーで表されます。

転送可能/転送済み
: 転送可能チケットはホストからホストに転送されます。これによって、クライアントは再び認証を受ける必要がありません。たとえば、ユーザー `david` がユーザー `jennifer` のマシンで転送可能チケットを取得した場合、david は自分のマシンにログインするときに新しいチケットを取得する必要はありません (再び認証を受けることもありません)。転送可能チケットの例については、例 26–1 を参照してください。

初期
: 初期チケットは、チケット認可チケットを使わずに直接発行されるチケットです。パスワードを変更するアプリケーションなどの一部のサービスでは、クライアントが非公開鍵を知っていることを確認するために、初期と指定されたチケットを要求することができます。初期チケットは、チケット認可チケットを使用せずに、クライ

	アントが最近認証されたことを証明します。チケット認可チケットの場合は、認証されてから時間が経っている可能性があります。
無効	無効チケットは、まだ使用可能になっていない遅延チケットです。無効チケットは、有効になるまでアプリケーションサーバーから拒否されます。これを有効にするには、開始時期が過ぎたあと、チケット認可サービス要求で VALIDATE フラグをオンにしてクライアントがこのチケットを KDC に提示する必要があります。
遅延可能/遅延	遅延チケットは、作成されても指定された時期まで有効にならないチケットです。たとえばこのようなチケットは、夜遅く実行するバッチジョブに使用するのに便利です。チケットが盗まれてもバッチジョブが実行される予定の時刻まで使用できないためです。遅延チケットは、無効チケットとして発行され、開始時刻を過ぎて、クライアントが KDC による検査を要求したときに有効になります。遅延チケットは通常、チケット認可チケットの有効期限まで有効です。ただし、チケットに更新可能が指定されている場合、その最長有効期限は通常、チケット認可チケットの最長有効期限と同じに設定されます。
プロキシ可能/プロキシ	場合によっては、サービスがそれ自身のために操作できることが主体にとって必要な場合があります。チケットを作成するときには、プロキシの主体名を指定する必要があります。Oracle Solaris リリースでは、プロキシ可能またはプロキシチケットをサポートしていません。 プロキシ可能チケットは転送チケットに似ていますが、プロキシ可能チケットが 1 つのサービスに対してだけ有効であることに対し、転送可能チケットはサービスに対しクライアントの識別情報の完全な使用を許可します。したがって、転送可能チケットは一種のスーパープロキシと考えられます。
更新可能	チケットに非常に長い有効期限を与えるとセキュリティーを損なうおそれがあるため、チケットを「更新可能」にすることができます。更新可能チケットには 2 つの有効期限があります。 1 つはチケットの現在のインスタンスの有効期限で、もう 1 つは任意のチケットの最長有効期限 (1 週間) です。クライアントがチケットの使用を継続するときは、最初の有効期限が切れる前にチケットの有効期限を更新します。たとえば、すべてのチケットの最長有効期限が 10 時間のときに、あるチケット

が 1 時間だけ有効だとします。このチケットを保持するクライアントが 1 時間を超えて使用する場合は、その時間内にチケットの有効期限を更新する必要があります。チケットが最長有効期限 (10 時間) に達すると、チケットの有効期限が自動的に切れ、それ以上更新できなくなります。

チケットの属性を表示する方法については、561 ページの「Kerberos チケットの表示」を参照してください。

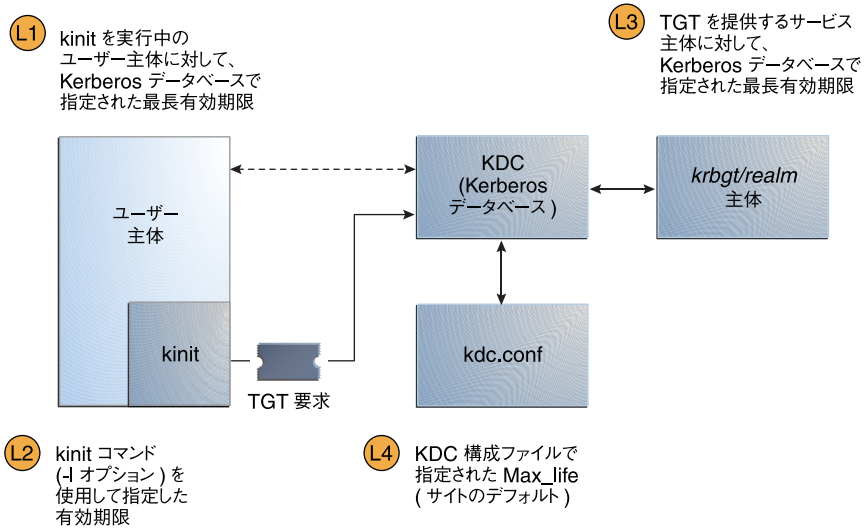
チケットの有効期限

主体がチケットを取得すると、チケット認可チケット (TGT) であっても、チケットの有効期限は次の中で最も小さい値に設定されます。

- `kinit` を使用してチケットを取得する場合、`kinit` の `-l` オプションに指定した有効期限値。デフォルトで、`kinit` は最長有効期限値を使用します。
- `kdc.conf` ファイルに指定されている最長有効期限値 (`max_life`)。
- チケットを提供するサービス主体に対し Kerberos データベースに指定されている最長有効期限値。`kinit` の場合、サービス主体は `krbtgt/`*realm*。
- チケットを要求するユーザー主体に対し Kerberos データベースに指定されている最長有効期限値。

図 27–1 は、TGT の有効期限の決定方法と 4 つの有効期限値の指定元を示しています。この図は TGT の有効期限がどのようにして決まるかを示しますが、基本的には、どの主体がチケットを取得する場合でも同じです。違いは、`kinit` で有効期限値を指定しないことと、`krbtgt/`*realm* 主体の代わりに、チケットを提供するサービス主体が最長有効期限値を提供することだけです。

図 27–1　TGT の有効期限の決定方法



チケットの有効期限 = L1、L2、L3、L4 の最小値

更新可能チケットの有効期限も次の 4 つの最小値で決まります。ただし、この場合は更新可能有効期限が使用されます。

- `kinit` を使用してチケットを取得または更新する場合、`kinit` の `-r` オプションに指定した更新可能有効期限値。
- `kdc.conf` ファイルに指定された更新可能最長有効期限値 (`max_renewable_life`)。
- チケットを提供するサービス主体に対し Kerberos データベースに指定されている更新可能最長有効期限値。`kinit` の場合、サービス主体は `krbtgt/`*realm*。
- チケットを要求するユーザー主体に対し Kerberos データベースに指定されている更新可能最長有効期限値。

Kerberos 主体名

チケットは主体名で識別され、主体名はユーザーやサービスを識別します。次の表に主体名の例を示します。

表 27–4　Kerberos 主体名の例

Principal Name	説明
`changepw/kdc1.example.com@EXAMPLE.COM`	パスワードを変更するときに、KDC にアクセスできるマスター KDC の主体。

表 27–4 Kerberos 主体名の例 (続き)

Principal Name	説明
clntconfig/admin@EXAMPLE.COM	kclient インストールユーティリティーで使用される主体。
ftp/boston.example.com@EXAMPLE.COM	ftp サービスによって使用される主体。この主体は host 主体の代わりに使用できます。
host/boston.example.com@EXAMPLE.COM	Kerberos アプリケーション (klist、 kprop など) およびサービス (ftp、 telnet など) によって使用される主体。この主体は host またはサービス主体と呼ばれます。主体は NFS マウントの認証に使用されます。この主体はまた、クライアントが受け取った TGT が正しい KDC から発行されたものであることを確認するためにも使用されます。
K/M@EXAMPLE.COM	マスター鍵名の主体。各マスター KDC には、1 つのマスター鍵名の主体が関連付けられます。
kadmin/history@EXAMPLE.COM	この主体に含まれる鍵を使用して、ほかの主体のパスワード履歴が保管されます。各マスター KDC には、これらの主体のいずれかが割り当てられます。
kadmin/kdc1.example.com@EXAMPLE.COM	kadmind を使用して KDC にアクセスできるマスター KDC サーバーの主体。
kadmin/changepw.example.com@EXAMPLE.COM	Oracle Solaris リリースが動作していないクライアントからのパスワード変更要求の受け入れに使用される主体。
krbtgt/EXAMPLE.COM@EXAMPLE.COM	この主体を使用して、チケット認可チケットを生成します。
krbtgt/EAST.EXAMPLE.COM@WEST.EXAMPLE.COM	この主体は、レルム間チケット認可チケットの例です。
nfs/boston.example.com@EXAMPLE.COM	NFS サービスによって使用される主体。この主体は host 主体の代わりに使用できます。
root/boston.example.com@EXAMPLE.COM	クライアントの root アカウントに関連付けられた主体。この主体は、root 主体と呼ばれ、NFS がマウントされたファイルシステムへの root アクセスを提供します。
username@EXAMPLE.COM	ユーザー用の主体。
username/admin@EXAMPLE.COM	KDC データベースを管理するために使用できる admin 主体。

Kerberos 認証システムの動作方法

アプリケーションを使用して遠隔システムにログインするには、識別情報を証明するチケットとそれに対応するセッション鍵を指定する必要があります。セッション鍵には、ユーザーやアクセスするサービスに特有の情報が含まれています。ユーザーすべてのチケットとセッション鍵は、ユーザーが最初にログインするときに KDC によって作成されます。チケットとそれに対応するセッション鍵が 1 つの資格となります。複数のネットワークサービスを使用する場合には、ユーザーは多数の資格を収集できます。ユーザーは特定のサーバーで動作するサービスごとに 1 つの資格を必要とします。たとえば、`boston` というサーバー上の `ftp` サービスにアクセスするには 1 つの資格が必要です。別のサーバー上の `ftp` サービスにアクセスするには、別の資格が必要です。

資格の作成や格納は透過的に行われます。資格は KDC によって作成され、要求者に送信されます。資格は、受信されると資格キャッシュに格納されます。

Kerberos サービスによる DNS および nsswitch.conf ファイルとの対話処理方法

Kerberos サービスは、ホスト名の解決に DNS を使用するようにコンパイルされています。ホスト名を解決する場合、 `nsswitch.conf` ファイルへの参照は一切行われません。

Kerberos によるサービスへのアクセス

特定のサーバー上の特定のサービスにアクセスする場合、ユーザーは 2 つの資格を取得する必要があります。最初は TGT として知られるチケット認可チケットに対する資格です。チケット認可サービスは、この資格の暗号を解除すると、ユーザーからアクセスを要求されているサーバーの資格をさらに作成します。ユーザーは、この 2 つめの資格を使用してサーバー上のサービスへのアクセスを要求します。サーバーがこの資格の暗号を解除すると、ユーザーはアクセスを許可されます。次の節では、このプロセスを詳細に説明します。

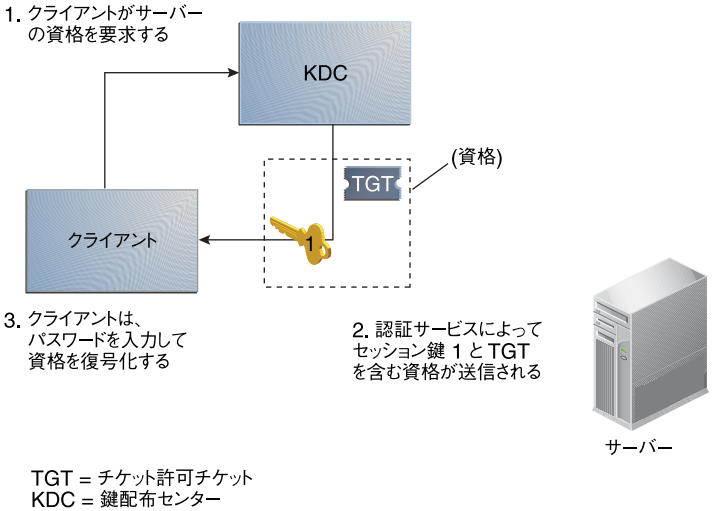
チケット認可サービスに対する資格の取得

1. 認証処理を開始するために、クライアントが特定のユーザー主体の要求を認証サーバーに送信します。この要求の送信では暗号は使用されません。要求にはセキュリティーにかかわる情報が含まれていないため、暗号を使う必要はありません。

2. 認証サービスは要求を受信すると、ユーザーの主体名を KDC データベースから検索します。主体がデータベースのエントリに一致すると、認証サービスはその主体の非公開鍵を取得します。次に認証サービスは、クライアントとチケット認可サービスが使用するセッション鍵 (セッション鍵 1) とチケット認可サービス用のチケット (チケット 1) を生成します。このチケットを「チケット認可チケット (TGT)」ともいいます。セッション鍵とチケットはユーザーの非公開鍵を使って暗号化され、情報がクライアントに返送されます。
3. クライアントは、ユーザー主体の非公開鍵を使用して、この情報からセッション鍵 1 とチケット 1 の暗号を解除します。非公開鍵を知っているのはユーザーと KDC データベースだけである必要があるため、パケットの情報は安全に保たれなければなりません。クライアントはこの情報を資格キャッシュに格納します。

この処理中に、ユーザーは通常、パスワードを要求されます。非公開鍵を作成するために使用された、KDC データベースに格納されているパスワードが、ユーザーが指定したパスワードと同じであると、認証サービスから送信された情報は正しく復号化されます。これでクライアントは、チケット認可サービスに対して使用する資格を取得します。次にクライアントはサーバーに対する資格を要求します。

図 27–2　チケット認可サービスに対する資格の取得



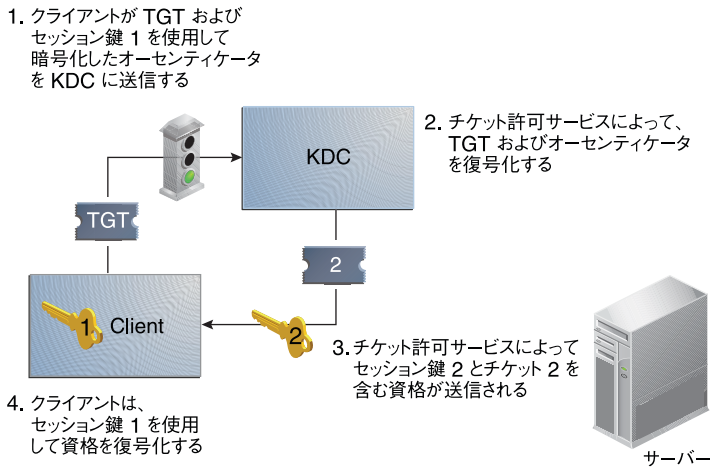
サーバーに対する資格の取得

1. 特定のサーバーにアクセスするには、クライアントがその前にサーバーに対する資格を認証サービスから取得している必要があります。587 ページの「チケット認可サービスに対する資格の取得」を参照してください。次にクライアントは、チケット認可サービスに要求を送信します。この要求には、サービス主体

名、チケット 1 およびセッション鍵 1 で暗号化されたオーセンティケータが含まれています。チケット 1 は、チケット認可サービスのサービス鍵を使用して認証サービスによって暗号化されたものです。

2. チケット認可サービスはチケット認可サービスのサービス鍵を知っているため、チケット 1 の暗号を解除できます。チケット 1 の情報にはセッション鍵 1 が含まれているため、チケット認可サービスはオーセンティケータの暗号を解除できます。この時点で、ユーザー主体はチケット認可サービスによって認証されます。
3. 認証が正常に終了すると、チケット認可サービスは、ユーザー主体とサーバーに対するセッション鍵 (セッション鍵 2) とサーバーに対するチケット (チケット 2) を生成します。次にセッション鍵 2 とチケット 2 はセッション鍵 1 を使って暗号化されます。セッション鍵 1 を知っているのはクライアントとチケット認可サービスだけであるため、この情報は安全であり、ネットワークを介して安全に送信されます。
4. クライアントはこの情報パケットを受信すると、前に資格キャッシュに格納したセッション鍵 1 を使用して情報を復号化します。クライアントは、サーバーに対して使用する資格を取得したことになります。次にクライアントは、そのサーバーの特定のサービスにアクセスする要求を行います。

図 27–3　サーバーに対する資格の取得

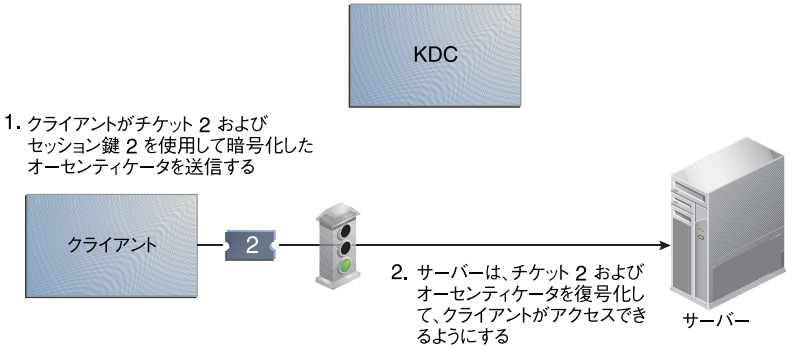


TGT = チケット許可チケット
KDC = 鍵配布センター

特定のサービスへのアクセス権の取得

1. クライアントが特定のサービスへのアクセスを要求するには、まず認証サーバーからチケット認可サービスに対する資格を取得し、チケット認可サービスからサーバー資格を取得する必要があります。587 ページの「チケット認可サービスに対する資格の取得」および 588 ページの「サーバーに対する資格の取得」を参照してください。次に、クライアントは、チケット 2 と別のオーセンティケータを含む要求をサーバーに送信します。オーセンティケータはセッション鍵 2 を使用して暗号化されます。
2. チケット 2 は、サービスのサービス鍵を使用してチケット認可サービスによって暗号化されています。サービス鍵はサービス主体が知っているため、サービスはチケット 2 を復号化し、セッション鍵 2 を取得できます。次に、セッション鍵 2 を使用してオーセンティケータが復号化されます。オーセンティケータが正しく復号化されると、サービスへのアクセスがクライアントに許可されます。

図 27–4 特定のサービスへのアクセス権の取得



Kerberos 暗号化タイプの使用

暗号化タイプは、暗号処理が実行されるときに使用する暗号アルゴリズムとモードを特定します。`aes`、`des3-cbc-sha1`、および `rc4–hmac` 暗号化タイプによって、より強固な暗号処理のために使用される鍵を作成できます。これらの強固な操作により、Kerberos サービスのセキュリティー全体が向上します。

注 – Solaris 10 8/07 より前のリリースでは、別売の Strong Cryptographic パッケージがインストールされている場合は、Kerberos サービスで `aes256-cts-hmac-sha1-96` 暗号化タイプを使用できます。

クライアントが KDC にチケットを要求する場合、KDC はクライアントとサーバーで互換性のある暗号化タイプの鍵を使用する必要があります。Kerberos プロトコルで

は、KDC がチケット応答のクライアント部分に対して特定の暗号化タイプを使用するようクライアントが要求することができます。しかし、サーバーが KDC に対して暗号化タイプを指定することはできません。

注 – Solaris 10 が動作していないマスター KDC がインストールされている場合は、マスター KDC をアップグレードする前に、スレーブ KDC を Solaris 10 にアップグレードする必要があります。Solaris 10 のマスター KDC では、古いスレーブが処理できない新しい暗号化タイプが使用されます。

次に、暗号化タイプを変更する前に考慮する必要があるいくつかの問題を説明します。

- KDC では、主体データベースのサーバー主体エントリに関連する最初の鍵/暗号化タイプはサーバーがサポートしていると想定します。
- KDC 上で、主体に対して生成される鍵が、主体が認証されるシステムと互換性があるかを確認してください。デフォルトで、`kadmin` コマンドは、サポートされるすべての暗号化タイプの鍵を生成します。主体を使用するシステムがこの暗号化タイプのデフォルトのセットをサポートしていない場合、主体の作成時に暗号化タイプを制限する必要があります。暗号化タイプは、`kadmin addprinc` で `-e` フラグを使用するか、または `kdc.conf` ファイルの `supported_enctypes` パラメータを設定することで、そのサブセットに制限することができます。Kerberos レルムの大部分のシステムがデフォルトの暗号化タイプのサブセットをサポートしている場合に、`supported_enctypes` パラメータを使用します。`supported_enctypes` の設定では、 `kadmin addprinc` が特定のレルムに対して主体を作成するときに使用する暗号化タイプのデフォルトのセットが指定されます。一般的に、これら 2 つの方法のうち 1 つを使用して、Kerberos が使用する暗号化タイプを制御します。
- システムがサポートする暗号化タイプを決定するときは、システムで実行する Kerberos のバージョンと、サーバー主体の作成対象のサーバーアプリケーションがサポートする暗号アルゴリズムとを考慮します。たとえば、`nfs/hostname` サービス主体を作成するときは、ホスト上の NFS サーバーがサポートするタイプに暗号化タイプを制限します。Solaris 10 リリースでは、サポートされるすべての Kerberos 暗号化タイプが NFS サーバーでもサポートされることに注意してください。
- `kdc.conf` ファイルの `master_key_enctype` パラメータを使用して、主体データベースのエントリを暗号化するマスター鍵の暗号化タイプを制御できます。KDC 主体データベースが既に作成済みの場合は、このパラメータを使用しないでください。データベースの作成時に `master_key_enctype` パラメータを使用して、デフォルトのマスター鍵の暗号化タイプを `des-cbc-crc` からより強固な暗号化タイプに変更できます。スレーブ KDC を設定するときに、すべてのスレーブ KDC が選択された暗号化タイプをサポートし、それらの `kdc.conf` に同一の `master_key_enctype` エントリがあることを確認してください。また、`supported_enctypes` が `kdc.conf` で設定されている場合は、`master_key_enctype` が `supported_enctypes` の暗号化タイプのうちの 1 つに設

定されていることも確認してください。これらの問題のいずれかが適切に処理されない場合、マスター KDC はスレーブ KDC と連携できない可能性があります。

- クライアント上で、krb5.conf のパラメータのいくつかを使用して、KDC からチケットを取得するときにクライアントが要求する暗号化タイプを制御できます。default_tkt_enctypes パラメータには、クライアントが KDC からチケット認可チケット (TGT) を要求するときに使用する暗号化タイプを指定します。TGT は、より効果的な方法でほかのサーバーチケットを取得するためにクライアントにより使用されます。default_tkt_enctypes を設定すると、クライアントが TGT を使用してサーバーチケットを要求する (TGS 要求と呼ばれる) ときに、クライアントと KDC 間の通信を保護するために使用される暗号化タイプを一部制御できます。default_tkt_enctypes に指定された暗号化タイプは、KDC 上に格納された主体データベース内の主体鍵の暗号タイプのうち少なくとも 1 つに一致する必要があることに注意してください。一致しない場合、TGT 要求は失敗します。多くの場合、default_tkt_enctypes を設定しないようにします。このパラメータは、相互運用性の問題の原因になるためです。デフォルトで、クライアントコードは、サポートされるすべての暗号化タイプと KDC が、KDC により主体データベースで検出された鍵に基づく暗号化タイプを選択するように要求します。
- default_tgs_enctypes パラメータは、サーバーチケットの取得に使用される TGS 要求でクライアントが要求する暗号化タイプを制限します。このパラメータは、クライアントとサーバーが共有するセッション鍵を作成するときに KDC が使用する暗号化タイプも制限します。たとえば、NFS をセキュリティー保護するときに 3DES 暗号化だけを使用するには、default_tgs_enctypes = des3-cbc-sha1 と設定します。クライアントとサーバーの主体が、主体データベース内に des-3-cbc-sha1 鍵を持っていることを確認してください。default_tkt_enctype と同様に、多くの場合でこの設定をしないようにします。資格が KDC とサーバーの両方で適切に設定されていないと、相互運用性の問題の原因となるためです。
- サーバー上で、kdc.conf の permitted_enctypes を使用して、サーバーが受け入れる暗号化タイプを制御できます。さらに、keytab エントリを作成するときに使用される暗号化タイプを指定できます。一般的に、これらの方法のいずれかを使用して暗号化タイプを制御したり、代わりに、使用する暗号化タイプを KDC に決定させたりしないようにします。KDC はサーバーアプリケーションと通信しないで、使用する鍵や暗号化タイプを決定するためです。

gsscred テーブルの使用

デフォルトのマッピングでは十分でないとき、NFS サーバーは gsscred テーブルを使用して、Kerberos ユーザーを識別します。NFS サービスは、UNIX ID を使用してユーザーを識別します。UNIX ID は、ユーザー主体または資格には含まれません。gsscred テーブルは、パスワードファイルから得られる GSS 資格を UNIX ID に追加してマッピングするテーブルです。このテーブルは、KDC データベースを生成し

たあとに作成および開始する必要があります。詳細は、424 ページの「GSS 資格の UNIX 資格へのマッピング」を参照してください。

クライアントの要求が到着すると、NFS サービスは主体名を UNIX ID にマッピングしようとします。このマッピングに失敗した場合、`gsscred` テーブルが確認されます。

Oracle Solaris Kerberos と MIT Kerberos の大きな違い

Solaris 10 バージョンの Kerberos サービスは MIT Kerberos バージョン 1.2.1 に基づいています。次に、Solaris 10 リリースに含まれ、MIT 1.2.1 バージョンには含まれない拡張機能を示します。

- Oracle Solaris 遠隔アプリケーションの Kerberos サポート
- KDC データベースの増分伝播
- クライアント構成スクリプト
- 地域対応のエラーメッセージ
- BSM 監査レコードのサポート
- GSS-API を使用する Kerberos のスレッドに対して安全な使用法
- 暗号用暗号化フレームワークの使用

このバージョンには MIT 1.2.1 以後のバグ修正もいくつか含まれています。特に、1.2.5 btree のバグ修正および 1.3 TCP サポートが追加されました。

パート VII

Oracle Solaris 監査

この節では、Oracle Solaris 監査サブシステムの設定、管理、および使用方法について説明します。

◆ ◆ ◆ 第 28 章

Oracle Solaris 監査 (概要)

Oracle Solaris 監査はシステムの使用状況を記録します。監査サービスには、監査データの分析を支援するツールが含まれています。

この章では、Oracle Solaris での監査機能について説明します。この章の内容は次のとおりです。

計画の提案については、第 29 章「Oracle Solaris 監査の計画」を参照してください。ご使用のシステムで監査を設定する手順については、第 30 章「Oracle Solaris 監査の管理 (手順)」を参照してください。参照情報については、第 31 章「Oracle Solaris 監査 (参照)」を参照してください。

監査とは

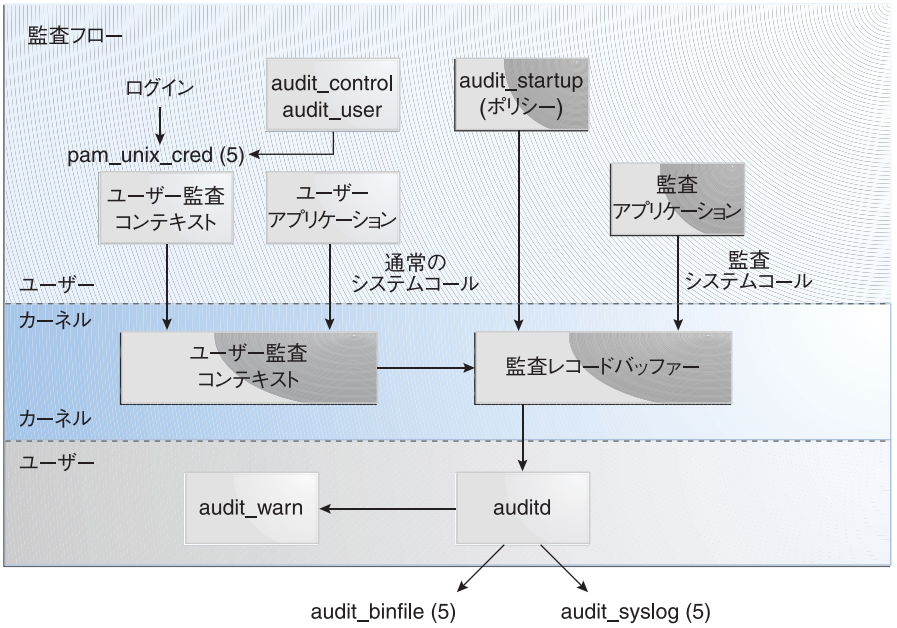
監査とは、システムリソースの使用状況に関するデータを収集することです。監査データは、セキュリティーに関連するシステムイベントの記録を提供します。このデータは、ホストで発生する動作に対する責任の割り当てに使用できます。監査を正常に機能させるには、識別と認証という 2 つのセキュリティー機能が重要 です。ログインのたびに、ユーザーがユーザー名とパスワードを入力すると、一意の監査セッション ID が生成され、そのユーザーのプロセスに関連付けられます。監査セッション ID は、ログインセッション中に起動されるすべてのプロセスに継承されます。単一のセッション内でユーザーが ID を変更しても、実行するすべての動作は同じ監査セッション ID によって追跡されます。ID の変更についての詳細は、su(1M) のマニュアルページを参照してください。

監査サービスを使うと、次の処理が可能になります。

- ホスト上で発生するセキュリティーに関係するイベントの監視
- ネットワーク全体の監査トレールへのイベントの記録
- 誤った使用または権限のない動作の検出
- アクセスパターンの確認と、ユーザーおよびオブジェクトのアクセス履歴の調査
- 保護メカニズムを迂回しようとする操作の検出
- ユーザーが ID を変更するときに発生する特権の拡大使用の検出

システムの構成時に、どの監査レコードのクラスを監視するかを選択します。各ユーザーに行う監査の程度は、細かく調整することもできます。次の図は、Oracle Solaris 監査のフローの詳細を示しています。

図 28–1　監査のフロー



監査データは、カーネルに収集された後、プラグインにより適切な場所に配布されます。次に、事後選択ツールで、監査証跡の必要な部分を削減して調査できます。たとえば、個々のユーザーまたは特定グループの監査レコードを確認できます。特定の日に発生した特定の種類のイベントに対するすべてのレコードを調査できます。または、特定の時間帯に生成されたレコードを選択することもできます。

非大域ゾーンをインストールするシステムは、大域ゾーンからと同じようにすべてのゾーンを監査できます。これらのシステムは、非大域ゾーンのさまざまなレ

コードを収集するように構成することもできます。詳細は、688 ページの「監査と Oracle Solaris ゾーン」を参照してください。

監査の機能

監査では、指定したイベントが発生したときに監査レコードが生成されます。通常、次のイベントが発生すると監査レコードが生成されます。

- システムの起動とシャットダウン
- ログインとログアウト
- プロセスまたはスレッドの作成と破棄
- オブジェクトを開く、閉じる、削除する、または名前を変更する
- 特権機能または役割に基づくアクセス制御 (RBAC) の使用
- 識別動作と認証動作
- プロセスまたはユーザーによるアクセス権の変更
- パッケージのインストールなどの管理動作
- サイト固有のアプリケーション

次の 3 つが監査レコードの生成元になります。

- アプリケーション
- 非同期監査イベントの結果
- プロセスのシステムコールの結果

関連するイベント情報が取得されると、その情報は監査レコードの書式に変換されます。次に、レコードは監査ファイルに書き込まれます。完全な監査レコードがバイナリ形式で格納されます。Solaris 10 リリースでは、`syslog` ユーティリティーを使用して、監査レコードを記録することもできます。

バイナリ形式の監査ファイルは、ローカルファイルシステムに格納できます。ファイルは、NFS がマウントされたファイルサーバーにも格納できます。監査ファイルの配置先として、同一システム上の複数のパーティション、異なる複数のシステム上のパーティション、または異なるが接続されているネットワーク上の複数のシステム上のパーティションを選択できます。接続された監査ファイルの集合は、監査トレールと呼ばれます。監査レコードは、発生順に監査ファイルに蓄積されます。各監査レコードには、イベントを識別する情報、イベントの発生元、イベントの時刻、およびその他の関連情報が格納されます。

監査レコードは、`syslog` ユーティリティーを使用して監視することもできます。これらの監査ログはローカルに格納することができます。あるいは、ログは、UDP プロトコルを使用して遠隔システムに送信することもできます。詳細は、605 ページの「監査ログ」を参照してください。

監査とセキュリティーとの関連

Oracle Solaris 監査を使用して、システムが通常と異なる方法で使用されていないかどうか検査すると、潜在的なセキュリティー侵害を検出できます。また、Oracle Solaris 監査によって、通常と異なる動作を追跡して、特定のユーザーを突き止めることができるため、セキュリティー侵害を抑止できます。活動を監査されていることをユーザーが知っている場合、悪質な活動を試みる可能性は低くなると考えられます。

コンピュータシステム、特にネットワーク上のシステムを保護するためには、システムのプロセスまたはユーザーのプロセスが開始する前に活動を制御するメカニズムが必要です。セキュリティーの確保には、動作の経過を監視するツールが必要となります。また、セキュリティーの確保には、動作終了後に動作内容を報告することも必要です。初期設定の Oracle Solaris 監査の場合、ユーザーのログイン前かシステムのプロセスが開始する前に、パラメータが設定されている必要があります。また、ほとんどの監査には、現在のイベントを監視し、指定されたパラメータを満たすイベントを報告する機能が含まれます。どのように Oracle Solaris 監査がこれらのイベントを監視し報告するかについては、第 29 章「Oracle Solaris 監査の計画」および第 30 章「Oracle Solaris 監査の管理 (手順)」を参照してください。

監査では、ハッカーによる不正な侵入を防止することはできません。ただし、監査サービスでは、たとえば、特定のユーザーが特定の日時に特定の動作を行ったことが報告されます。監査報告では、入力経路とユーザー名によってこのユーザーを特定できます。これらの情報は、すぐに端末に表示したり、ファイルに出力してあとで分析したりできます。このように、監査サービスの提供するデータから、次のことが判断できます。

- どのようにシステムセキュリティーが低下したか
- 必要なセキュリティーレベルを実現するために閉じることが必要なセキュリティーホールはどれか

監査の用語と概念

監査サービスでは、次の用語が使用されています。定義によっては、より詳細な説明への参照先も示します。

表 28–1 Oracle Solaris 監査用語

用語	定義
監査クラス	監査イベントのグループ。監査クラスは、監査対象のイベントのグループの選択方法を提供します。詳細は、603 ページの「監査クラスおよび事前選択」を参照してください。

表 28–1 Oracle Solaris 監査用語 (続き)

用語	定義
監査ディレクトリ	バイナリ形式の監査ファイルのリポジトリ。監査ディレクトリのタイプについては、605 ページの「監査ログ」を参照してください。
監査イベント	監査対象のセキュリティー関連のシステム動作。選択を容易にするために、イベントは監査クラスにグループ化されます。監査対象のシステム動作については、602 ページの「監査イベント」を参照してください。
監査ポリシー	サイトで有効または無効を指定できる監査オプションのセット。特定の種類の監査データを記録するかどうかのオプションがあります。また、監査トレールがいっぱいになった場合に監査を中断するかどうかも指定できます。詳細は、617 ページの「監査ポリシーの決定」を参照してください。
監査レコード	監査ファイルに格納される監査データ。1 つの監査レコードにつき 1 つの監査イベントが記述されます。各監査レコードは、一連の監査トークンから構成されます。監査レコードの詳細は、604 ページの「監査レコードと監査トークン」を参照してください。
監査トークン	監査レコードまたはイベントのフィールド。各監査トークンには、監査イベントの 1 つの属性 (ユーザー、プログラム、その他のオブジェクトなど) が記述されます。すべての監査トークンの説明は、696 ページの「監査トークンの形式」を参照してください。
監査トレール	監査サービスを実行するすべてのシステムからの監査データを格納する 1 つまたは複数の監査ファイルの集合。詳細は、693 ページの「監査トレール」を参照してください。
事前選択	事前選択とは、監査サービスの有効化前に行う、監視する監査クラスの選択です。事前選択された監査クラスの監査イベントは、監査トレールに記録されます。事前選択されない監査クラスは監査されず、監査トレールに記録されません。事後選択ツールの `auditreduce` コマンドを使用すると、監査トレールからレコードを選択できます。詳細は、603 ページの「監査クラスおよび事前選択」を参照してください。
公開オブジェクト	公開オブジェクトとは、`root` ユーザーが所有し、すべてのユーザーが読み取り可能なファイルです。たとえば、`/etc` ディレクトリと `/usr/bin` ディレクトリは公開オブジェクトです。公開オブジェクトの読み取り専用イベントは監査されません。たとえば、`file_read (fr)` 監査フラグが事前選択されている場合でも、公開オブジェクトの読み取り動作は監査されません。`public` 監査ポリシーオプションを変更すると、デフォルトを上書きできます。
監査プラグイン	カーネルキューの監査レコードを指定した場所に転送するモジュール。`audit_binfile.so` プラグインはバイナリ監査ファイル (監査証跡) を作成します。`audit_syslog.so` プラグインは、選択した監査レコードを `syslog` ログにフィルタします。詳細は、604 ページの「監査プラグインモジュール」を参照してください。

監査イベント

セキュリティーに関連するシステム動作について監査を行うことができます。これらの監査可能な動作を、「監査イベント」と呼びます。監査イベントは、/etc/security/audit_event ファイルに指定します。各監査イベントは、このファイル内で、イベント番号、シンボル名、簡単な説明、およびそのイベントが属する一連の監査クラスによって定義されます。audit_event ファイルの詳細は、audit_event(4) のマニュアルページを参照してください。

たとえば、次のエントリは、exec() システムコールの監査イベントを定義します。

```
7:AUE_EXEC:exec(2):ps,ex
```

監査クラス ps または ex の監査を事前選択すると、exec() システムコールは監査トレールに記録されます。

Oracle Solaris 監査は、「ユーザーに起因する」イベントと「ユーザーに起因しない」イベントを処理します。監査ポリシーでは、イベントが「同期イベント」と「非同期イベント」に分割されます。次のようになります。

- ユーザーに起因するイベント – ユーザーによって起こるイベント。exec() システムコールはユーザーに起因します。そのため、この呼び出しはユーザーに起因するイベントと見なされます。ユーザーに起因するイベントはすべて、同期イベントです。
- ユーザーに起因しないイベント – カーネル割り込みレベルで、またはユーザーが認証される前に発生するイベント。na 監査クラスは、ユーザーに起因しない監査イベントを処理します。たとえば、システムのブートはユーザーに起因しないイベントです。ユーザーに起因しないイベントの多くは、非同期イベントです。ただし、失敗したログインなど、プロセスが関連付けられている場合、ユーザーに起因しないイベントは同期イベントです。
- 同期イベント – システムのプロセスに関連付けられたイベント。システムイベントの大半は同期イベントです。
- 非同期イベント – プロセスに関連付けられていないイベント。そのため、ブロックした後に呼び起こすプロセスはありません。たとえば、システムの初期起動や PROM の開始および終了のイベントは、非同期イベントです。

監査イベントが属するクラスが事前選択されていると、イベントは監査トレールに記録されます。たとえば、監査対象に ps と na 監査クラスを事前選択すると、ほかのイベントとともに exec() システムコールおよびシステムのブート動作が監査トレールに記録されます。

Oracle Solaris 監査サービスによって定義される監査イベントのほかに、Sun 以外のアプリケーションも監査イベントを生成できます。32768 から 65535 までの監査イベント番号が Sun 以外のアプリケーションで使用できます。

監査クラスおよび事前選択

各監査イベントは、1 つまたは複数の「監査クラス」に属しています。監査クラスは、多数の監査イベントが入った便利な入れ物です。監査対象としてクラスを「事前選択」すると、そのクラスのすべてのイベントが監査トレールに記録されるように指定されます。システム上のイベントまたは特定のユーザーによって開始されるイベントに対して事前選択できます。監査サービスの実行開始後、事前選択されたクラスに監査クラスを動的に追加したり削除したりできます。

- システム全体の事前選択 – `audit_control` ファイルの `flags`、 `naflags`、 および `plugin` 行にシステム全体の監査デフォルトを指定します。`audit_control` ファイルについては、682 ページの「`audit_control` ファイル」を参照してください。また、`audit_control(4)` のマニュアルページも参照してください。
- ユーザー固有の事前選択 – `audit_user` データベースにある個々のユーザーに対するシステム全体の監査デフォルトに追加を指定します。

  監査事前選択マスクにより、ユーザーに対して監査されるイベントのクラスを決定します。ユーザーの監査事前選択マスクとは、システム全体のデフォルトとユーザーに対して指定された監査クラスの組み合わせです。詳細については、693 ページの「プロセスの監査特性」を参照してください。

  `audit_user` データベースは、ローカルまたはネームサービスで管理できます。Solaris 管理コンソールは、データベースを管理するグラフィカルユーザーインタフェース (GUI) を備えています。詳細は、`audit_user(4)` のマニュアルページを参照してください。
- 動的な事前選択 – プロセスまたはセッションに追加したり削除したりする監査クラスを `auditconfig` コマンドの引数として指定します。詳細は、`auditconfig(1M)` のマニュアルページを参照してください。

事後選択コマンド `auditreduce` を使用すると、事前選択された監査レコードからレコードを選択できます。詳細は、607 ページの「監査トレールの検証」および `auditreduce(1M)` のマニュアルページを参照してください。

監査クラスは、`/etc/security/audit_class` ファイルに定義されます。各エントリには、クラスの監査マスク、クラスの名前、およびクラスの記述名が含まれます。たとえば、`ps` および `na` クラス定義は、次のように `audit_class` に表示されます。

```
0x00100000:ps:process start/stop
0x00000400:na:non-attribute
```

監査クラスは、32 クラスまで設定できます。クラスには、`all` および `no` という 2 つのグローバルクラスが 含まれます。監査クラスについては、`audit_class(4)` のマニュアルページを参照してください。

監査イベントのクラスへの割り当ては構成可能です。クラスからイベントを削除したり、クラスにイベントを追加したり、新しいクラスを作成して選択したイベントを含めることができます。手順については、633 ページの「監査イベントの所属先クラスの変更方法」を参照してください。

監査レコードと監査トークン

各「監査レコード」には、監査された 1 つのイベントの発生が記録されます。レコードには、動作を行ったユーザー、影響を受けたファイル、試みられた動作、その動作が発生した位置と時刻などの情報が含まれます。次の例は、login 監査レコードの例です。

```
header,81,2,login - local,,2003-10-13 11:23:31.050 -07:00
subject,root,root,other,root,other,378,378,0 0 example_system
text,successful login
return,success,0
```

監査イベントごとに保存される情報の種類は、一連の「監査トークン」によって定義されます。イベントの監査レコードが生成されるたびに、そのイベントに対して定義されたトークンの一部またはすべてが、そのレコードに書き込まれます。どのトークンが記録されるかは、イベントの性質によって決まります。前の例では、各行が監査トークンの名前で始まっています。監査トークンの内容は、名前のあとに続きます。4 つの監査トークンが集まって、login 監査レコードを構成しています。

praudit 出力例の各監査トークンの構造については、696 ページの「監査トークンの形式」を参照してください。監査トークンのバイナリストリームについては、audit.log(4) のマニュアルページを参照してください。

監査プラグインモジュール

監査プラグインモジュールを指定して、事前選択で監査キューに配置したレコードを処理できます。プラグインは、audit_control ファイル内のエントリです。

- audit_binfile.so プラグイン - バイナリ監査ファイルへの監査キューの配信を処理します。audit_control ファイルで、プラグインが指定されておらず dir エントリに値がある場合、監査デーモンがこのプラグインを使用します。
- audit_syslog.so プラグイン - 選択したレコードの監査キューから syslog ログへの配信を処理します。

audit_control ファイルの構文は、audit_control(4) のマニュアルページを参照してください。たとえば、624 ページの「監査ファイルの構成 (作業マップ)」でタスクを参照します。

プラグインについては、audit_binfile(5)、 audit_syslog(5)、 および audit_control(4) のマニュアルページを参照してください。

監査ログ

監査レコードは監査ログ内に収集されます。Oracle Solaris 監査では、監査ログ用の 2 つの出力モードがあります。「監査ファイル」と呼ばれるログは、バイナリ形式で監査レコードを格納します。システムまたはサイトからの一連の監査ファイルでは、完全な監査レコードが提供されます。完全な監査レコードは「監査トレール」と呼ばれます。

`syslog` ユーティリティーを使用して、テキストバージョンの監査レコードの概要を収集して格納します。`syslog` レコードは、完全なレコードではありません。次の例は、`login` 監査レコードの `syslog` エントリを示します。

```
Oct 13  11:24:11 example_system auditd: [ID 6472 audit.notice] \
        login - login ok session 378 by root as root:other
```

サイトは監査レコードを両方の形式で格納できます。使用中のシステムを設定して、バイナリモード、`syslog` モード、または両方のモードを使用できます。次の表は、バイナリ形式の監査レコードと `syslog` 監査レコードの比較を示します。

表 **28–2** バイナリ形式の監査レコードと syslog 監査レコードの比較

機能	バイナリレコード	**syslog** レコード
プロトコル	ファイルシステムに書き込みます	遠隔ログ記録に UDP を使用します
データ型	バイナリ形式	テキスト
レコード長	無制限	監査レコードあたり最大 1024 文字
場所	ローカルディスク、および NFS を使用してマウントされたディレクトリに格納されます	`syslog.conf` ファイルで指定された場所に格納されます
設定方法	`audit_control` ファイルを編集して、監査ディレクトリを保護し NFS でマウントします	`audit_control` ファイルを編集して、`syslog.conf` ファイルを編集します
読み取り方法	通常、バッチモード XML 形式によるブラウザの出力	リアルタイムで、または `syslog` 用に作成したスクリプトによる検索 プレーンテキスト形式の出力
完全性	完全であることと正確な順番で表示されることが保証されます	完全であることは保証されません
タイムスタンプ	グリニッジ標準時 (GMT)	監査されているシステム上の時間

バイナリ形式のレコードにより、強力なセキュリティーとカバレージが提供されます。バイナリ出力は、共通基準制御アクセス保護プロファイル (CAPP) などのセキュリティー認証の要件を満たします。各レコードは、傍受されないように保護したファイルシステムに書き込まれます。 単一のシステム上で、すべてのバイナリ形

式のレコードが、収集され順番に表示されます。バイナリログの GMT タイムスタンプにより、特定の監査トレールのシステムが複数の時間帯にわたって配信されるとき、正確に比較することができます。praudit -x コマンドを使用すると、レコードを XML 形式でブラウザに表示できます。スクリプトを使用して、XML を解析することもできます。

これに対し、syslog レコードは、より扱いやすく柔軟性が優れています。たとえば、さまざまなソースから syslog データを収集できます。また、syslog.conf ファイル内の audit.notice イベントを監視するとき、syslog ユーティリティーを使用して、現在のタイムスタンプで監査レコードの概要のログをとることができます。ワークステーション、サーバー、ファイアウォール、ルーターなどのさまざまなソースからの syslog メッセージ用に開発された管理および分析ツールと同じツールを使用できます。レコードをリアルタイムで表示し、遠隔システム上に格納することができます。

syslog.conf を使用して監査レコードを遠隔に格納すると、攻撃者による改変や削除からログデータが保護されます。その一方で、監査レコードを遠隔に格納すると、レコードは、サービス拒否や偽装されたソースアドレスなどのネットワーク攻撃を受けやすくなります。また、UDP はパケットを破棄したり、間違った順番でパケットを配信したりすることがあります。syslog エントリの制限は、1024 文字です。そのため、一部の監査レコードがログで切り捨てられる可能性があります。単一のシステム上で、すべての監査レコードが収集されるわけではありません。レコードが順番に表示されない場合もあります。各監査レコードにはローカルシステムの日時が記録されているため、タイムスタンプを基に、複数のシステム用の監査トレールを構成することはできません。

監査ログについての詳細は、次を参照してください。

- audit_syslog(5) のマニュアルページ
- audit.log(4) のマニュアルページ
- 627 ページの「syslog 監査ログの構成方法」

監査証跡の格納

「監査ディレクトリ」はバイナリ形式で監査ファイルを保持します。通常のインストールでは、多くの監査ディレクトリが使用されます。すべての監査ディレクトリの内容は、「監査トレール」で構成されています。監査レコードは、次の順番で監査ディレクトリに格納されます。

- 「一次監査ディレクトリ」 – 通常の条件下で、システム監査ファイルが配置されるディレクトリ
- 「二次監査ディレクトリ」 – 一次監査ディレクトリがいっぱいか使用できない場合に、システム監査ファイルが配置されるディレクトリ
- 「最後のディレクトリ」 – 一次監査ディレクトリと二次監査ディレクトリが使用できない場合に使用されるローカル監査ディレクトリ

ディレクトリは、audit_control ファイルに指定されます。各ディレクトリは、上記の順番で前のディレクトリがいっぱいになるまで使用されません。ディレクトリエントリのリストを含む注釈付きの audit_control ファイルについては、例 30–3 を参照してください。

監査ファイルをデフォルトの監査ルートディレクトリに配置すると、監査証跡を確認する際に役立ちます。auditreduce コマンドは、監査ルートディレクトリを使用して監査証跡内のすべてのファイルを検索します。デフォルトの監査ルートディレクトリは /etc/security/audit です。このディレクトリは、/var/audit に象徴的にリンクされています。/var/audit/*hostname*/files という名前のディレクトリにある監査ファイルは、auditreduce コマンドで簡単に見つかります。詳細は、677 ページの「auditreduce コマンド」を参照してください。

監査トレールの検証

監査サービスには、監査トレールのファイルを結合したり削除したりするコマンドがあります。auditreduce コマンドは、監査トレールの監査ファイルをマージします。また、ファイルをフィルタして特定のイベントを検出できます。praudit コマンドは、バイナリファイルを読み取ります。praudit コマンドのオプションにより、スクリプトやブラウザの表示に適した出力が得られます。

Oracle Solaris ゾーンを含むシステムでの監査

ゾーンは、Oracle Solaris OS の単一インスタンス内に作成される仮想化オペレーティング環境です。監査サービスは、ゾーン内でのアクティビティーも含め、システム全体を監査します。非大域ゾーンがインストールされたシステムでは、単一の監査サービスを実行してすべてのゾーンを同様に監査することができます。あるいは、大域ゾーンも含め、ゾーンごとに監査サービスを 1 つずつ構成することもできます。

次の条件を満たすサイトでは、単一の監査サービスを実行できます。

- 単一イメージの監査トレールを必要とするサイトである。
- 非大域ゾーンがアプリケーションコンテナとして使用されている。ゾーンが 1 つの管理ドメインの一部になっています。つまり、どの非大域ゾーンにおいても、ネームサービスファイルがカスタマイズされていません。

  システム上のすべてのゾーンが 1 つの管理ドメインに含まれている場合、zonename 監査ポリシーを使えば、複数のゾーンで実行される監査イベントを区別できます。
- 管理者が低い監査オーバーヘッドを望んでいる。大域ゾーン管理者がすべてのゾーンを同様に監査します。また、大域ゾーンの監査デーモンがシステム上のすべてのゾーンを処理します。

次の条件を満たすサイトでは、ゾーンごとに監査サービスを 1 つずつ実行できます。

- 単一イメージの監査トレールを必要としないサイトである。
- 非帯域ゾーンでネームサービスファイルがカスタマイズされている。これらの個々の管理ドメインは通常、サーバーとして機能します。
- 個々のゾーン管理者が、自身が管理するゾーンの監査を制御する必要がある。ゾーンごとの監査では、ゾーン管理者は、自身が管理するゾーンの監査を有効にするか無効にするかを決定できます。

ゾーンごとの監査の利点は、監査トレールをゾーンごとにカスタマイズできることと、ゾーンの監査をゾーン単位で無効化できることです。これらの利点は、管理オーバーヘッドによって相殺される可能性があります。ゾーン管理者はすべての監査構成ファイルをカスタマイズします。各ゾーンでは、独自の監査デーモンが実行され、独自の監査キューや監査ログが用意されます。ゾーンの監査ログファイルの管理を行う必要があります。

Solaris 10 リリースでの 監査拡張機能

Solaris 9 リリースと比較して、次の機能が 監査に追加されました。

- 監査で、`syslog` ユーティリティーを使用して監査ファイルをテキスト形式で格納できます。詳細は、605 ページの「監査ログ」を参照してください。`syslog` ユーティリティーを使用するために `audit_control` ファイルを設定する方法は、627 ページの「`syslog` 監査ログの構成方法」を参照してください。
- `praudit` コマンドで、新しい出力形式 (XML) が使用できるようになりました。XML は、移植性のある、処理可能な標準の形式です。XML 形式の出力は、ブラウザを使用して表示できるほか、報告用に XML スクリプトへの入力としても使えます。 `praudit` コマンドの `-x` オプションについては、679 ページの「`praudit` コマンド」を参照してください。
- 一連のデフォルトの監査クラスが整理されました。監査メタクラスによって、より細かい監査クラスをグループ化することができます。クラスのデフォルト集合の一覧については、689 ページの「監査クラスの定義」を参照してください。
- `bsmconv` コマンドを使用しても、Stop-A キーの使用が無効化されなくなりました。Stop-A イベントは監査可能です。
- 監査レコードのタイムスタンプは、ISO 8601 形式で報告されます。この標準については、`http://www.iso.org` を参照してください。
- 3 つの監査ポリシーオプションが追加されました。
  - **public –** 読み取り専用イベントの公開オブジェクトは監査されなくなりました。公開ファイルを監査しないようにすれば、監査ログの量を大幅に減らすことができます。このため、機密性の高いファイルの読み取りイベントが監視しやすくなります。公開オブジェクトについては、600 ページの「監査の用語と概念」を参照してください。

- **perzone** – perzone ポリシーは広く影響します。別々の監査デーモンが各ゾーンで実行します。デーモンは、ゾーンに固有の監査構成ファイルを使用します。また、監査キューもゾーンに固有です。詳細については、auditd(1M) および auditconfig(1M) のマニュアルページを参照してください。ゾーンの詳細は、688 ページの「監査と Oracle Solaris ゾーン」を参照してください。ポリシーの詳細は、612 ページの「ゾーン内の監査の計画方法」を参照してください。
- **zonename** – 監査イベントが発生した Oracle Solaris ゾーンの名前が監査レコードに含まれます。ゾーンの詳細は、688 ページの「監査と Oracle Solaris ゾーン」を参照してください。オプションを使用する場合については、617 ページの「監査ポリシーの決定」を参照してください。

- 5 つの監査トークンが追加されました。
  - cmd トークンは、コマンドに割り当てられた引数のリストおよび環境変数のリストを記録します。詳細は、700 ページの「cmd トークン」を参照してください。
  - path_attr トークンは、path トークンオブジェクトの下にある属性ファイルオブジェクトの処理を記録します。詳細は、707 ページの「path_attr トークン」を参照してください。
  - privilege トークンは、プロセス上の特権の使用を記録します。詳細は、707 ページの「privilege トークン」を参照してください。
  - uauth トークンは、コマンドまたは動作による承認の使用を記録します。詳細は、714 ページの「uauth トークン」を参照してください。
  - zonename トークンは、監査イベントが発生した非大域ゾーンの名前を記録します。zonename 監査ポリシーのオプションにより、zonename トークンが監査レコードに含まれるかどうかを指定します。詳細は、715 ページの「zonename トークン」を参照してください。

    参照情報については、688 ページの「監査と Oracle Solaris ゾーン」を参照してください。ゾーンの詳細は、『Oracle Solaris のシステム管理 (Oracle Solaris コンテナ : 資源管理と Oracle Solaris ゾーン)』のパート II「ゾーン」を参照してください。

◆ ◆ ◆ 第 29 章

Oracle Solaris 監査の計画

この章では、インストールした Oracle Solaris に対して監査サービスを設定する方法について説明します。特に、監査サービスを有効にする前に、考慮する必要のある問題について説明します。この章の内容は次のとおりです。

監査の概要は、第 28 章「Oracle Solaris 監査 (概要)」を参照してください。ご使用のシステムで監査を設定する手順については、第 30 章「Oracle Solaris 監査の管理 (手順)」を参照してください。参照情報については、第 31 章「Oracle Solaris 監査 (参照)」を参照してください。

Oracle Solaris 監査の計画 (作業マップ)

次の作業マップは、ディスク容量および記録するイベントの計画に必要とされる主要な作業を示しています。

作業	参照先
非大域ゾーンの監査計画を決定します	612 ページの「ゾーン内の監査の計画方法」
監査トレールの記憶領域容量を計画します	613 ページの「監査レコード用の記憶領域を計画する方法」
監査対象者と監査対象イベントを決定します	614 ページの「監査対象者と監査対象イベントの計画方法」

Oracle Solaris 監査の計画 (手順)

監査する動作の種類を適切に選択し、有用な監査情報を収集することが望まれます。監査ファイルはすぐに大きくなり、空き領域がなくなる可能性があるため、十分なディスク容量を割り当てる必要があります。監査対象者と監査対象も注意して計画する必要があります。

▼ ゾーン内の監査の計画方法

システムでゾーンが実装されている場合、監査を構成する方法が 2 つあります。

- すべてのゾーンに対して大域ゾーン内で単一の監査サービスを構成できます。
- ゾーンごとに監査サービスを 1 つずつ構成できます。

トレードオフについては、607 ページの「Oracle Solaris ゾーンを含むシステムでの監査」を参照してください。

- 次のいずれかの方法を選択します。

  - **オプション 1 - すべてのゾーンに対して単一の監査サービスを構成します。**

    すべてのゾーンを同様に監査する場合、単一イメージの監査トレールが作成される可能性があります。単一イメージの監査トレールが作成されるのは、システム上のすべてのゾーンが 1 つの管理ドメインの一部になっている場合です。その場合、すべてのゾーン内のレコードが同一の設定に基づいて事前選択されるため、監査レコードの比較が容易に行えます。

    この構成では、すべてのゾーンが 1 つのシステムの一部として扱われます。大域ゾーンがシステム上で唯一の監査デーモンを実行して、すべてのゾーンに対して監査ログを収集します。監査構成ファイルのカスタマイズを大域ゾーン内でのみ行ったあと、それらの監査構成ファイルをすべての非大域ゾーンにコピーします。

    a. **`audit_control`** ファイルを大域ゾーンからすべての非大域ゾーンにコピーします。

    b. すべてのゾーンで同じ **`audit_user`** データベースを使用します。

    `audit_user` データベースは、ローカルファイルでも、共有されたネームサービスから取得してもかまいません。

    c. ゾーンに基づいて監査レコードを選択できるようにします。

    ゾーン名を監査レコードの一部として含めるには、大域ゾーンで `zonename` ポリシーを設定します。次に、`auditreduce` コマンドで、監査証跡からゾーンに基づいて監査イベントを選択できます。例については、`auditreduce(1M)` のマニュアルページを参照してください。

単一イメージ監査証跡を計画するには、614 ページの「監査対象者と監査対象イベントの計画方法」を参照してください。最初の手順から始めます。また、大域ゾーン管理者は、613 ページの「監査レコード用の記憶領域を計画する方法」の説明に従って記憶領域を確保する必要もあります。

- **オプション 2 - ゾーンごとに監査サービスを 1 つずつ構成できます。**

  ゾーンごとにネームサービスファイルが異なる場合や、ゾーン管理者が自身のゾーン内の監査を制御しようとする場合には、ゾーンごとの監査の構成を選択します。

  - ゾーンごとの監査を構成する場合は、大域ゾーンに監査を構成する必要があります。大域ゾーンで `perzone` 監査ポリシーを設定します。監査ポリシーを設定するには、649 ページの「ゾーンごとの監査を構成する方法」を参照してください。

    注 – ネームサービスファイルが非大域ゾーンでカスタマイズされ、`perzone` ポリシーが設定されていない場合は、使用可能なレコードの選択に監査ツールを注意して使用する必要があります。あるゾーン内のユーザー ID は、異なるゾーン内の同じ ID とは異なるユーザーを指している可能性があります。

  - 発生元のゾーンを追跡できるレコードを生成するには、大域ゾーンで `zonename` 監査ポリシーを設定します。大域ゾーン内で、`zonename` を使用して `auditreduce` コマンドを実行します。次に、`zonename` ゾーン内で、`auditreduce` の出力に対して `praudit` コマンドを実行します。
  - 各ゾーン管理者がゾーンの監査ファイルを構成します。

    非大域ゾーン管理者は、`perzone` と `ahlt` 以外のすべてのポリシーオプションを設定できます。
  - 各ゾーン管理者はゾーン内で監査を有効化または無効化できます。

  すべてのゾーン内の監査構成ファイルをカスタマイズするには、614 ページの「監査対象者と監査対象イベントの計画方法」を参照して、すべてのゾーン用に計画します。最初の手順は省略できます。また、各ゾーン管理者は、613 ページの「監査レコード用の記憶領域を計画する方法」の説明に従って各ゾーンの記憶領域を確保する必要もあります。

▼ 監査レコード用の記憶領域を計画する方法

監査トレールには専用のファイル領域が必要です。監査ファイル用の専用ファイル領域は、使用可能でセキュリティー保護されている必要があります。各システムには、監査ファイル用に構成されたいくつかの監査ディレクトリが必要です。システ

ム上で監査を有効にする前に、最初に監査ディレクトリの構成方法を決定する必要があります。次の手順は、監査トレール記憶領域を計画するときに、解決する必要のある問題を扱っています。

始める前に　非大域ゾーンを実装している場合は、この手順を行う前に612 ページの「ゾーン内の監査の計画方法」の手順を完了してください。

1. **サイトで必要な監査の程度を決定します。**

   サイトのセキュリティー上の必要性を考慮して、監査トレールに使用できるディスク容量を決定します。

   サイトのセキュリティーを確保しながら領域要件を削減する方法と、監査記憶領域を設定する方法については、620 ページの「監査コストの制御」と622 ページの「効率的な監査」を参照してください。

2. **監査対象のシステムを決定します。**

   これらのシステム上で、少なくとも 1 つのローカル監査ディレクトリに容量を割り当てます。監査ディレクトリを指定する方法は、例 30–3 を参照してください。

3. **監査ファイルを格納するシステムを決定します。**

   一次および二次監査ディレクトリを保持するサーバーを決定します。監査ディレクトリ用のディスクの構成例は、635 ページの「監査ファイルのパーティションの作成方法」を参照してください。

4. **監査ディレクトリの名前をつけます。**

   使用するすべての監査ディレクトリの一覧を作成します。命名ガイドラインについては、606 ページの「監査証跡の格納」および677 ページの「auditreduce コマンド」を参照してください。

5. **システムと監査ディレクトリの対応付けを決定します。**

   システムと監査ディレクトリのマッピングを作成します。このマッピングにより、監査作業の負荷を分散します。図については、図 31–1 と図 31–2 を参照してください。

▼ 監査対象者と監査対象イベントの計画方法

始める前に　非大域ゾーンを実装している場合は、この手順を行う前に612 ページの「ゾーン内の監査の計画方法」の手順を完了してください。

1 単一システムイメージの監査トレールが必要かどうかを決定します。

システムが単一の管理ドメイン内にある場合、単一システムイメージの監査トレールを作成できます。各システムが異なるネームサービスを使用している場合は、次の手順から始めます。すべてのシステムごとに計画の手順を完了する必要があります。

単一システムイメージ監査トレールは監査対象のシステムを 1 つのマシンとして扱います。サイト用の単一システムイメージ監査トレールを作成するには、インストール内のすべてのシステムを次のように構成する必要があります。

- 同一のネームサービスを使用する。

  監査レコードを解釈するには、`auditreduce` と `praudit` の 2 つのコマンドを使用します。監査レコードを正しく解釈するためには、`passwd`、 `hosts`、 および `audit_user` ファイルの整合性がとれている必要があります。
- すべてのシステムに対して、同じ `audit_warn`、 `audit_event`、 `audit_class`、 および `audit_startup` ファイルを使用します。
- 同一の `audit_user` データベースを使用します。このデータベースは、NIS や LDAP などのネームサービス内に存在していてもかまいません。
- `audit_control` ファイル内の `flags`、 `naflags`、 および `plugin` エントリを同じにします。

2 監査ポリシーを決定します。

`auditconfig -lspolicy` コマンドを使用して、使用可能なポリシーオプションを表示します。デフォルトでは、`cnt` ポリシーだけが使用可能になっています。詳細は、手順 8 を参照してください。

ポリシーオプションの働きについては、617 ページの「監査ポリシーの決定」を参照してください。監査ポリシーを設定するには、639 ページの「監査ポリシーを構成する方法」を参照してください。

3 イベントからクラスへのマッピングを変更するかを決定します。

多くの場合、デフォルトのマッピングをそのまま使用できます。ただし、新しいクラスの追加、クラス定義の変更、あるいは、特定のシステムコールのレコードが役に立たないという判断をした場合は、イベントを別のクラスに移動する必要がある場合もあります。

例は、633 ページの「監査イベントの所属先クラスの変更方法」を参照してください。

4 事前選択する監査クラスを決定します。

監査クラスの追加やデフォルトクラスの変更は、監査サービスを開始する前に行うことを推奨します。

audit_control ファイル内の flags、 naflags、 および plugin エントリの監査クラス値は、すべてのユーザーとプロセスに適用されます。事前選択されたクラスによって、監査クラスが監査されるのは成功の場合か、失敗の場合か、またはその両方の場合かが決定されます。

監査クラスを事前選択するには、625 ページの「audit_control ファイルの変更方法」を参照してください。

5 システム全体の事前選択された監査クラスのユーザー例外を決定します。

一部のユーザーの監査をシステム全体の事前選択された監査クラスと異なる設定にするときは、audit_user データベース内の各ユーザーのエントリを変更します。

例は、630 ページの「ユーザーの監査特性の変更方法」を参照してください。

6 最小空きディスク容量を決定します。

監査ファイルシステム上のディスク容量が minfree の割合を下回ると、auditd デーモンは次に利用できる監査ディレクトリに切り替えます。デーモンは、弱い制限値を超えたことを示す警告を送信します。

最小空きディスク容量を設定するには、例 30–4 を参照してください。

7 **audit_warn** 電子メールのエイリアスの管理方法を決定します。

audit_warn スクリプトは、監査システムが管理上の注意を要求する状況を通知する必要があるときに実行されます。デフォルトでは、audit_warn スクリプトは、audit_warn のエイリアスに電子メールを送信し、コンソールにメッセージを送信します。

エイリアスを設定するには、639 ページの「audit_warn 電子メールエイリアスの構成方法」を参照してください。

8 すべての監査ディレクトリがいっぱいになったときの動作を決定します。

デフォルトでは、監査トレールのオーバーフローが発生しても、システムは引き続き動作します。システムは破棄された監査レコードを数えますが、イベントを記録しません。セキュリティーをより向上させるためには、cnt ポリシーを無効にして、ahlt ポリシーを有効にします。ahlt ポリシーは、非同期イベントが監査キューに配置できない場合、システムを停止します。

これらのポリシーオプションについては、619 ページの「非同期イベントおよび同期イベントの監査ポリシー」を参照してください。これらのポリシーオプションを設定するには、例 30–16 を参照してください。

9 監査レコードを、バイナリ形式、**syslog** 形式、または両方の形式のいずれで収集するかを決定します。

概要は、605 ページの「監査ログ」を参照してください。

例は、627 ページの「syslog 監査ログの構成方法」を参照してください。

監査ポリシーの決定

監査ポリシーを使用して、ローカルシステムの監査レコードの特性を決定します。監査ポリシーオプションは、起動スクリプトによって設定されます。監査サービスを有効にする bsmconv スクリプトによって、/etc/security/audit_startup スクリプトが作成されます。audit_startup スクリプトは、auditconfig コマンドを実行することで監査ポリシーを設定します。スクリプトの詳細は、audit_startup(1M) のマニュアルページを参照してください。

ほとんどの監査ポリシーオプションがデフォルトで無効になっているのは、記憶領域要件とシステム処理要求を最小限に抑えるためです。監査ポリシーオプションを動的に有効または無効にするには、auditconfig コマンドを使用します。監査ポリシーオプションを永続的に有効または無効にするには、audit_startup スクリプトを使用します。

次の表を参照して、1 つまたは複数の監査ポリシーオプションを有効にしたときに発生する追加のオーバーヘッドを考慮しながら、サイトの要件を決定してください。

表 29–1 監査ポリシーオプションの働き

ポリシー名	説明	ポリシーオプションを変更する理由
ahlt	非同期イベントにだけ適用されます。無効にすると、監査レコードが生成されないまま、イベントを完了できます。 有効にすると、監査ファイルシステムがいっぱいになるとシステムを停止します。監査キューの再配置、監査レコードの空き容量の確保、および再起動は管理者の介入が必要です。大域ゾーンでだけ有効にできます。ポリシーはすべてのゾーンに影響します。	セキュリティーよりシステムの可用性が重要な場合は、無効にします。 セキュリティーを最優先する場合は、有効にします。
arge	無効にすると、実行されたプログラムの環境変数が exec 監査レコードから除外されます。 有効にすると、実行されたプログラムの環境変数が exec 監査レコードに追加されます。監査レコードには、より詳細な情報が記録されます。	無効にすると、収集される情報が大幅に少なくなります。 このオプションは、少数のユーザーを監査するときに有効にします。このオプションは、exec プログラムで使用される環境変数に問題があるときにも有用です。
argv	無効にすると、実行されたプログラムの引数が exec 監査レコードから除外されます。 有効にすると、実行されたプログラムの引数が exec 監査レコードに追加されます。監査レコードには、より詳細な情報が記録されます。	無効にすると、収集される情報が大幅に少なくなります。 このオプションは、少数のユーザーを監査するときに有効にします。このオプションは、exec プログラムが正常に動作しないことがはっきりしているときにも有用です。

表 29–1 監査ポリシーオプションの働き (続き)

ポリシー名	説明	ポリシーオプションを変更する理由
cnt	無効にすると、ユーザーまたはアプリケーションの実行がブロックされます。このブロックが発生するのは、空きディスク容量の不足により監査トレールに監査レコードが追加できない場合です。 有効にすると、監査レコードが生成されないまま、イベントを完了できます。生成されなかった監査レコードのカウントは行われます。	セキュリティーを最優先する場合は、無効にします。 セキュリティーよりシステムの可用性が重要な場合は、有効にします。
group	無効にすると、グループの一覧が監査レコードに追加されません。 有効にすると、グループの一覧が特別なトークンとしてすべての監査レコードに追加されます。	通常は無効にしてもサイトのセキュリティー要件は満たします。 どのグループが監査イベントを生成しているかを監査する必要があるときは、有効にします。
path	無効にすると、1 つのシステムコールで使用されたパスが、あっても 1 つだけ監査レコードに記録されます。 有効にすると、監査イベントで使用されたすべてのパスが、すべての監査レコードに記録されます。	無効にすると、監査レコードにパスが、あっても 1 つだけ記録されます。 有効にすると、1 つのシステムコールで使用された各ファイル名またはパスが、監査レコードに path トークンとして記録されます。
perzone	無効にすると、システムの単一の監査構成を保守します。大域ゾーン内で 1 つのデーモンが実行されます。zonename 監査トークンを事前選択すると、非大域ゾーンの監査イベントは、監査レコード内に置かれます。 有効にすると、各ゾーンの監査構成、監査キュー、および監査ログを別々に保守します。単独バージョンの監査デーモンが各ゾーンで実行されます。大域ゾーンでだけ有効にできます。	各ゾーンごとに監査ログ、キュー、およびデーモンを保守する理由が特にない場合は、無効にすると便利です。 単に zonename 監査トークンを事前選択することではシステムを効果的に監視できない場合は、有効にすると便利です。
public	無効にすると、ファイルの読み取りが事前に選択されている場合に、公開オブジェクトの読み取り専用イベントが監査トレールに追加されなくなります。読み取り専用イベントを含む監査クラスとしては、fr、 fa、 および cl があります。 有効にすると、適切な監査フラグが事前に選択されている場合、公開オブジェクトの読み取り専用監査イベントのすべてが記録されます。	通常は無効にしてもサイトのセキュリティー要件は満たします。 このオプションを有効にするのはまれです。
seq	無効にすると、すべての監査レコードに順序番号が追加されません。 有効にすると、すべての監査レコードに順序番号が追加されます。順序番号は sequence トークンに格納されます。	監査が問題なく動作しているときは、無効にしても構いません。 cnt ポリシーが有効なときは、有効にする意味があります。seq ポリシーにより、いつデータが破棄されるかを決定できます。

表 29–1 監査ポリシーオプションの働き (続き)

ポリシー名	説明	ポリシーオプションを変更する理由
trail	無効にすると、trailer トークンが監査レコードに追加されません。	無効にすると、作成される監査レコードが小さくなります。
	有効にすると、trailer トークンがすべての監査レコードに追加されます。	有効にすると、各監査レコードの最後に trailer トークンが常に付加されます。trailer トークンは、多くの場合 sequence トークンと一緒に使用されます。trailer トークンを使用すると、監査レコードの再同期が容易で正確になります。
zonename	無効にすると、zonename トークンが監査レコードに含まれません。	ゾーン間で監査動作を比較する必要がない場合は、無効にすると便利です。
	有効にすると、zonename トークンが非大域ゾーンからのすべての監査レコードに含まれます。	ゾーン間で監査動作を特定し比較する場合は、有効にすると便利です。

非同期イベントおよび同期イベントの監査ポリシー

ahlt ポリシーおよび cnt ポリシーは、監査キューがいっぱいで追加のイベントを受け入れられない場合の動作を管理します。ポリシーは独立して関連しています。ポリシーの組み合わせには、それぞれ次のような効果があります。

- -ahlt +cnt は、出荷時のデフォルトのポリシーです。このデフォルトのポリシーにより、イベントが記録できない場合でも、監査対象イベントが処理されます。

  -ahlt ポリシーでは、非同期イベントの監査レコードがカーネル監査キューに配置できない場合、システムがイベントをカウントして処理を続行します。大域ゾーンで、as_dropped カウンタがカウントを記録します。

  +cnt ポリシーでは、同期イベントがカーネル監査キューに到達しても配置できない場合、システムがイベントをカウントして処理を続行します。ゾーンの as_dropped カウンタがカウントを記録します。

  -ahlt +cnt の構成は通常、処理の続行により監査レコードが失われる可能性があっても処理を続行する必要のある場合に使用します。auditstatdrop フィールドは、ゾーンで破棄された監査レコードの数を示します。

- +ahlt -cnt ポリシーでは、イベントがカーネル監査キューに追加できない場合、処理が停止します。

  +ahlt ポリシーでは、非同期イベントの監査レコードがカーネル監査キューに配置できない場合、すべての処理が停止されます。システムはパニック状態になります。非同期イベントは、監査キューには入らず、呼び出しスタックのポインタから復元する必要があります。

-cnt ポリシーでは、同期イベントがカーネル監査キューに配置できない場合、イベントを配信しようとするスレッドがブロックされます。スレッドは、監査領域が使用可能になるまでスリープキューに配置されます。カウントは保持されません。プログラムは、監査領域が使用可能になるまでハングアップしたように見えることがあります。

+ahlt -cnt の構成は通常、システムの可用性より監査イベントの記録を優先する場合に使用します。プログラムは、監査領域が使用可能になるまでハングアップしたように見えます。auditstat wblk フィールドは、スレッドがブロックされた回数を示します。

ただし、非同期イベントが発生した場合、システムがパニック状態になり停止します。監査イベントのカーネルキューは、保存したクラッシュダンプから手動で復元できます。非同期イベントは、監査キューには入らず、呼び出しスタックのポインタから復元する必要があります。

- -ahlt -cnt ポリシーでは、非同期イベントがカーネル監査キューに配置できない場合、イベントがカウントされ処理が続行します。同期イベントがカーネル監査キューに配置できない場合、イベントを配信しようとするスレッドがブロックされます。スレッドは、監査領域が使用可能になるまでスリープキューに配置されます。カウントは保持されません。プログラムは、監査領域が使用可能になるまでハングアップしたように見えることがあります。

  -ahlt -cnt の構成は通常、非同期監査レコードが失われる可能性より、すべての同期監査イベントの記録を優先する場合に使用します。auditstat wblk フィールドは、スレッドがブロックされた回数を示します。

- +ahlt +cnt ポリシーでは、非同期イベントがカーネル監査キューに配置できない場合、システムがパニック状態になります。同期イベントがカーネル監査キューに配置できない場合、システムがイベントをカウントして処理を続行します。

監査コストの制御

監査処理によってシステムリソースが消費されるため、どの程度詳しく記録するかを制御する必要があります。監査の対象を決めるときには、監査に伴う次の 3 つのコストを考慮してください。

- 処理時間の増大に伴うコスト
- 監査データの分析に伴うコスト
- 監査データの格納に伴うコスト

監査データの処理時間の増大に伴うコスト

処理時間の増大に伴うコストは、監査に関連する 3 つのコストの中ではもっとも重要性の低い問題です。第 1 に、通常は、イメージ処理や複雑な計算処理などの計算

集中型のタスクの実行中には、監査処理が発生しないからです。その他の理由として、単一ユーザーシステムのコストが通常は無視できるほど小さいことが挙げられます。

監査データの分析に伴うコスト

分析に伴うコストは、収集される監査データの量にほぼ比例します。分析コストには、監査レコードをマージして検討するための時間が含まれます。コストにはまた、監査レコードをアーカイブし、それを安全な場所に保管するための時間も含まれます。

生成されるレコードの数が少ないほど、監査トレールの分析にかかる時間も短くなります。次の節、621 ページの「監査データの格納に伴うコスト」と 622 ページの「効率的な監査」では、効率的に監査を行う方法について説明します。効果的な監査では、収集するデータの量を削減しながら、サイトのセキュリティー目標を達成します。

監査データの格納に伴うコスト

記憶領域コストは、監査コストのうちでもっとも重要です。監査データの量は次の要素によって左右されます。

- ユーザー数
- システム数
- 使用量
- 要求される追跡容易性と説明義務の程度

これらの要因はサイトごとに異なるため、監査データの格納用に前もって確保しておくディスク容量を決定できるような計算式はありません。次の情報を参考にしてください。

- 監査クラスを慎重に事前選択し、生成されるレコードの量を減らします。

  `all` クラスを指定して完全な監査を行うと、ディスクがすぐにいっぱいになります。プログラムのコンパイルといった単純なタスクによってさえ、巨大な監査ファイルが生成される可能性があります。中程度のサイズのプログラムでも、1 分も経たないうちに数千件の監査レコードが生成される可能性があります。

  たとえば、`file_read` 監査クラス `fr` を省くと、監査ボリュームを著しく削減できます。失敗した処理だけの監査を選択すると、監査ボリュームが減ることもあります。たとえば、失敗した `file_read` 処理 `-fr` のみを監査すると、すべての `file_read` イベントの監査よりも生成されるレコードを大幅に少なくすることができます。
- また、監査ファイルを効率的に管理することも重要です。監査レコードが生成されたあとにファイル管理を行うことによって、必要なディスク容量を減らせます。

- 監査クラスを理解します。

  監査を構成する前に、クラスに含まれるイベントの種類を理解しておく必要があります。監査のイベントからクラスへのマッピングを変更すると、監査レコード収集を最適化できます。
- サイトの監査方針を策定します。

  実用的な方法を基に方針を策定します。そのような基準には、サイトで必要な追跡容易性の程度や、管理対象ユーザーの種類などが含まれます。

効率的な監査

次の方法により、組織のセキュリティー目標を達成する一方で、監査効率を高めることができます。

- 一回にある一定の割合のユーザーのみをランダムに監査します。
- 監査ファイルのディスク容量要件を削減するために、監査ファイルを結合、縮小、および圧縮します。ファイルを保管する手順、リムーバブルメディアにファイルを転送する手順、およびファイルをオフラインで格納する手順を決定します。
- 監査データの異常な動作をリアルタイムで監視します。すでに持っている管理および分析ツールを拡張すると、`syslog` ファイル内の監査レコードを処理できます。

  また、特定の動作に対して監査トレールを監視する手順を設定します。異常なイベントが検出された場合に、それに応じて特定のユーザーまたは特定のシステムの監査レベルを自動的に上げるようなスクリプトを作成します。

  たとえば、次のようにスクリプトを作成します。

  1. すべての監査ファイルサーバー上における監査ファイルの作成を監視します。
  2. `tail` コマンドを使用して、監査ファイルを処理します。

     `tail -0f` コマンドから `praudit` コマンドに出力をパイプすることにより、レコードが生成されたときに監査レコードのストリームを生成できます。詳細は、`tail(1)` のマニュアルページを参照してください。
  3. このストリームを分析して異常なメッセージの種類やほかの兆候を調べ、または監査担当者に分析を配信します。

     また、このスクリプトを使用して、自動応答を発生させることもできます。
  4. 監査ディレクトリを常時監視して、新しい `not_terminated` 監査ファイルが発生していないかを調べます。
  5. 監査ファイルに書き込めなくなったときに、未処理の `tail` プロセスを終了します。

◆ ◆ ◆ 第 30 章

Oracle Solaris 監査の管理 (手順)

この章では、監査対象の Oracle Solaris システムの設定と管理に役立つ手順を説明します。また、監査トレールの管理方法についても説明します。この章の内容は次のとおりです。

監査サービスの概要は、第 28 章「Oracle Solaris 監査 (概要)」を参照してください。計画の提案については、第 29 章「Oracle Solaris 監査の計画」を参照してください。参照情報については、第 31 章「Oracle Solaris 監査 (参照)」を参照してください。

Oracle Solaris 監査 (作業マップ)

次の作業マップは、監査の管理に必要な主な作業を示します。作業は順番に並んでいます。

作業	説明	参照先
1. 監査を計画します	監査サービスを構成する前に判断する構成上の問題を含みます。	611 ページの「Oracle Solaris 監査の計画 (作業マップ)」
2. 監査ファイルを構成します	監査を必要とするイベント、クラス、およびユーザーを定義します。	624 ページの「監査ファイルの構成 (作業マップ)」

作業	説明	参照先
3. 監査を構成および有効化します	ディスク容量とほかの監査サービスの要件に対して各ホストを構成します。その後、監査サービスを開始します。	634 ページの「監査サービスの構成と有効化 (作業マップ)」
	非大域ゾーンがインストールされたホストでは、システムに対して 1 つの監査サービスを構成するか、ゾーンごとに監査サービスを 1 つずつ構成します。	646 ページの「ゾーンでの監査サービスの構成 (手順)」
4. 監査レコードを管理します。	監査データを収集して分析します。	650 ページの「監査レコードの管理 (作業マップ)」

監査ファイルの構成 (作業マップ)

次の作業マップは、サイトで監査をカスタマイズするファイルの構成手順の一覧です。ほとんどの作業は省略可能です。

作業	説明	参照先
監査クラスの選択、および `audit_control` 設定のカスタマイズを行います	次のことを行います: ■ システム全体の監査クラスを事前に選択する ■ システムごとに監査ディレクトリを指定する ■ 監査ファイルシステム上のディスク容量の制限を設定する	625 ページの「`audit_control` ファイルの変更方法」
(省略可能) 2 つのモードでの監査イベントを記録します	バイナリ形式で監査レコードを格納できるようにするほか、リアルタイムでの監視を有効にします。	627 ページの「`syslog` 監査ログの構成方法」
(省略可能) ユーザーの監査特性を変更します	システム全体の事前選択された監査クラスに対してユーザー固有の例外を設定します。	630 ページの「ユーザーの監査特性の変更方法」
(省略可能) 監査クラスを追加します	イベントを保持する新しい監査クラスを作成して、監査レコードの数を減らします。	632 ページの「監査クラスの追加方法」
(省略可能) イベントからクラスへのマッピングを変更します	イベントからクラスへのマッピングを変更して、監査レコードの数を減らします。	633 ページの「監査イベントの所属先クラスの変更方法」

監査ファイルの構成 (手順)

ネットワーク上で監査を有効にする前に、サイトの監査要件の監査構成ファイルをカスタマイズします。また、監査サービスを再起動またはローカルシステムをリブートして、監査サービスが有効になったあとに変更された構成ファイルの読み取りを行うこともできます。ただし、監査構成のカスタマイズに必要な作業は、できる限り監査サービスを開始する前に行います。

ゾーンを実装している場合、大域ゾーンからすべてのゾーンを監査することができます。監査出力内のゾーンを区別するには、zonename ポリシーオプションを指定します。非大域ゾーンを別々に監査するには、大域ゾーン内の perzone ポリシーを指定して、非大域ゾーンの監査構成ファイルをカスタマイズします。概要については、688 ページの「監査と Oracle Solaris ゾーン」を参照してください。計画については、612 ページの「ゾーン内の監査の計画方法」を参照してください。手順については、646 ページの「ゾーンでの監査サービスの構成 (手順)」を参照してください。

▼ audit_control ファイルの変更方法

/etc/security/audit_control ファイルを使用して、システム全体の監査を構成します。ファイルは、監査対象のイベント、監査警告の発行時期、および監査ファイルの場所を決定します。

1 **Primary Administrator 役割を引き受けるか、スーパーユーザーになります。**

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 **(省略可能) audit_control ファイルのバックアップコピーを保存します。**

```
# cp /etc/security/audit_control /etc/security/audit_control.orig
```

3 **サイトの audit_control ファイルを変更します。**

各エントリの書式は次のとおりです。

keyword:*value*

keyword 行の種類を定義します。種類には、dir、 flags、 minfree、 naflags、 および plugin があります。Solaris 10 リリースでは、dir 行および minfree 行は非推奨です。

キーワードの説明は、次の例を参照してください。

value その種類の行に関連するデータを指定します。

注 – 監査ディレクトリの場所を指定するには、audit_binfile.so プラグインの p_dir 属性を使用します。最小空き容量を指定するには、p_minfree 属性を使用します。

4 (省略可能) ファイルの構文を確認します。

```
# audit -v /etc/security/audit_control
syntax ok
```

例 30–1 すべてのユーザーの監査クラスを事前選択する

audit_control ファイルの flags 行には、監査対象のユーザーに起因するイベントのクラスを定義します。このフラグは、システム上のすべてのユーザーに適用されます。クラスをコンマで区切ります。空白が使用できます。この例では、すべてのユーザーを対象に lo クラスおよび ap クラスのイベントが監査されます。。

```
## audit_control file
flags:lo,ap
naflags:lo
plugin:name=...
```

どのイベントがクラスに割り当てられているかを確認するには、audit_event ファイルを参照してください。bsmrecord コマンドを使用することもできます (例 30–27 参照)。

例 30–2 ユーザーに起因しないイベントを事前選択する

この例では、na クラス内のすべてのイベント、およびユーザーに起因しないすべての login イベントが監査対象です。

```
## audit_control file
flags:lo
naflags:lo,na
plugin:name=...
```

例 30–3 バイナリ監査データの場所を指定する

audit_binfile.so プラグインの p_dir フラグは、バイナリ監査データに使用する監査ファイルシステムを一覧表示します。この例では、バイナリ監査データ用に 3 つの場所が定義されています。ディレクトリは、1 次ディレクトリから予備のディレクトリの順に一覧表示されます。plugin 行には改行は含まれません。

```
## audit_control file
##
flags:lo
naflags:lo,na
```

```
plugin:name=audit_binfile.so; p_dir=/var/audit/egret.1/files,
/var/audit/egret.2/files,/var/audit
```

バイナリ監査データを保持するファイルシステムの設定方法は、635 ページの「監査ファイルのパーティションの作成方法」を参照してください。

例 30–4 警告のための弱い制限値を変更する

この例では、すべての監査ファイルシステムの最小空き領域レベルが設定され、利用できるファイルシステムの領域が 10% だけになったときに警告が発行されるようにします。

plugin 行には改行は含まれません。

```
## audit_control file
#
flags:lo
naflags:lo,na
plugin:name=audit_binfile.so; p_dir=/var/audit/examplehost.1/files,
/var/audit/examplehost.2/files,/var/audit/localhost/files; p_minfree=10
```

audit_warn エイリアスが警告を受け取ります。エイリアスを設定するには、639 ページの「audit_warn 電子メールエイリアスの構成方法」を参照してください。

▼ syslog 監査ログの構成方法

監査サービスで、監査キューにある一部またはすべての収集した監査レコードを syslog にコピーできます。次の手順では、バイナリ監査データとテキスト監査データを保存します。収集されたテキスト監査データは、バイナリデータのサブセットです。

始める前に 監査クラスを事前に選択する必要があります。事前に選択される監査クラスは、audit_control ファイルの naflags 行および flags 行で指定します。また、audit_user ファイルの個々のユーザーについてクラスを事前に選択し、auditconfig コマンドで監査クラスを動的に追加できます。

1. **Primary Administrator 役割を引き受けるか、スーパーユーザーになります。**

   Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2. **(省略可能) audit_control ファイルのバックアップコピーを保存します。**

   ```
   # cp /etc/security/audit_control /etc/security/audit_control.save
   ```

3 **audit_syslog.so プラグインエントリを追加します。**

```
## audit_control file
flags:lo,ss
naflags:lo,na
plugin:name=audit_binfile.so;p_dir=/var/audit; p_minfree=20;
plugin:name=audit_syslog.so;p_flags=+lo,-ss
```

plugin エントリの書式は次のとおりです。

plugin:name=*name*; qsize=*max-queued-records*;p_*=*value*

- name= *name* – プラグインの名前を一覧表示します。有効な値は、audit_binfile.so および audit_syslog.so です。
- qsize= *max-queued-records* – プラグインに送信される監査データについて、キューに入れるレコードの最大数を指定します。この属性は省略可能です。
- p_*=*value* – プラグイン固有の属性を指定します。audit_syslog.so プラグインは p_flags を受け入れます。audit_binfile.so プラグインは、p_dir、 p_minfree、および p_fsize を受け入れます。p_fsize 属性は、Solaris 10 10/08 で導入されました。

プラグイン固有の属性の詳細は、audit_binfile(5) の OBJECT ATTRIBUTES の節および audit_syslog(5) のマニュアルページを参照してください。

4 **audit.notice エントリを syslog.conf ファイルに追加します。**

エントリは、ログファイルの場所を含みます。

```
# cat /etc/syslog.conf
...
audit.notice       /var/adm/auditlog
```

テキストログは、バイナリ監査ファイルの格納場所には格納しないでください。バイナリ監査ファイルを読み取る auditreduce コマンドは、監査パーティション内にあるファイルをすべてバイナリ監査ファイルと見なします。

5 ログファイルを作成します。

```
# touch /var/adm/auditlog
```

6 **syslog サービスの構成情報を更新します。**

```
# svcadm refresh system/system-log
```

7 **syslog ログファイルを定期的に保管します。**

監査サービスでは、大量の出力が生成される可能性があります。ログの管理方法については、logadm(1M) のマニュアルページを参照してください。

例 30–5 syslog 出力の監査クラスを指定する

次の例では、syslog ユーティリティーを使用して、事前に選択された監査クラスのサブセットを収集します。

```
## audit_user file
jdoe:pf

## audit_control file
flags:lo,ss
naflags:lo,na
plugin:name=audit_binfile.so; p_dir=/var/audit/host.1/files,
/var/audit/host.2/files,/var/audit/localhost/files; p_minfree=10
plugin:name=audit_syslog.so; p_flags=-lo,-na,-ss,+pf
```

flags と naflags エントリにより、システムはログインおよびログアウト、ユーザーに起因しないイベント、およびシステム状態の変更のすべての監査レコードをバイナリ形式で収集します。audit_syslog.so プラグインエントリにより、syslog ユーティリティーは失敗したログイン、ユーザーに起因しない失敗したイベント、および失敗したシステム状態の変更だけを収集します。jdoe ユーザーの場合、バイナリ監査レコードにはプロファイル認識シェルの使用がすべて含まれます。syslog ユーティリティーは、成功したプロファイル認識コマンドを収集します。pf クラスは、例 30–10 で作成されます。

例 30–6 syslog 監査レコードを遠隔システムに配置する

syslog.conf ファイル内の audit.notice エントリを変更して、遠隔システムを指定します。この例では、ローカルシステムの名前は、example1 です。遠隔システムは、remote1 です。

```
example1 # cat /etc/syslog.conf
...
audit.notice       @remote1
```

remote1 システム上の syslog.conf ファイル内にある audit.notice エントリはログファイルを指定します。

```
remote1 # cat /etc/syslog.conf
...
audit.notice       /var/adm/auditlog
```

例 30–7 audit_control ファイルでプラグインを使用する

audit_control ファイルのフラグなし情報を指定する際には、plugin エントリの使用をお勧めします。この例では、監査フラグの選択の後に、プラグイン情報が一覧表示されています。

```
## audit_control file
flags:lo,ss
naflags:lo,na
plugin:name=audit_binfile.so;p_minfree=10; p_dir=/var/audit
plugin:name=audit_syslog.so; p_flags=+lo
```

▼ ユーザーの監査特性の変更方法

ユーザーごとの定義は、audit_user データベースに格納されます。 これらの定義は、指定されたユーザーに関して、audit_control ファイルで事前に選択されたクラスを変更します。nsswitch.conf ファイルは、ローカルファイルまたはネームサービスデータベースのどちらを使用するかを決定します。ユーザーの最終監査事前定義マスクを計算する方法は、693 ページの「プロセスの監査特性」を参照してください。

1 **Primary Administrator 役割を引き受けるか、スーパーユーザーになります。**

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 **(省略可能) audit_user データベースのバックアップコピーを保存します。**

```
# cp /etc/security/audit_user /etc/security/audit_user.orig
```

3 **audit_user データベースに新しいエントリを追加します。**

ローカルデータベースの場合、各エントリの書式は次のようになります。

username:*always-audit*:*never-audit*

username	監査するユーザー名を選択します。
always-audit	指定したユーザーについて常に監査する監査クラスの一覧を選択します。
never-audit	指定したユーザーについて監査しない監査クラスの一覧を選択します。

複数のクラスを指定するには、監査クラスをコンマで区切ります。

audit_user エントリは、ユーザーの次回のログイン時に有効になります。

例 30–8 1 人のユーザーに関して監査するイベントを変更する

この例では、audit_control ファイルは、システム用の事前に選択された監査クラスを含みます。

```
## audit_control file
...
```

```
flags:lo,ss
naflags:lo,na
```

audit_user ファイルは例外を表示します。ユーザー jdoe がプロファイルシェルを使用すると、監査されます。

```
## audit_user file
jdoe:pf
```

jdoe の監査事前選択マスクは、audit_user 設定と audit_control 設定を結合したものです。auditconfig -getaudit コマンドは、jdoe の事前選択マスクを表示します。

```
# auditconfig -getaudit
audit id = jdoe(1234567)
process preselection mask = ss,pf,lo(0x13000,0x13000)
terminal id (maj,min,host) = 242,511,example1(192.168.160.171)
audit session id = 2138517656
```

例 30–9 システムではなくユーザーだけを監査する

この例では、システム上での 4 人のログインと役割活動が監査されます。audit_control ファイルは、システム用の監査クラスを事前に選択しません。

```
## audit_control file
...
flags:
naflags:
```

audit_user ファイルは、次のように 4 人のユーザー用の 2 つの監査クラスを事前に選択します。

```
## audit_user file
jdoe:lo,pf
kdoe:lo,pf
pdoe:lo,pf
sdoe:lo,pf
```

次の audit_control ファイルは、不当な侵入を記録します。audit_user ファイルとの組み合わせにより、この例の最初の audit_control ファイルに比べてシステムをより確実に保護します。

```
## audit_control file
...
flags:
naflags:lo
plugin:name=...
```

▼ 監査クラスの追加方法

独自の監査クラスを作成すると、サイトで監視する監査イベントだけをそのクラスに入れることができます。1 つのシステムにそのクラスを追加するときは、監査されているすべてのシステムに変更をコピーする必要があります。

1 **Primary Administrator 役割を引き受けるか、スーパーユーザーになります。**

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 **(省略可能) `audit_class` ファイルのバックアップコピーを保存します。**

```
# cp /etc/security/audit_class /etc/security/audit_class.orig
```

3 **`audit_class` ファイルに新しいエントリを追加します。**

各エントリの書式は次のとおりです。

`0x`*number*`:`*name*`:`*description*

0x	*number* が 16 進であることを示します。
number	一意の監査クラスマスクを定義します。
name	監査クラスの文字の名前を定義します。
description	監査クラスの記述名を定義します。

エントリはファイル内で一意である必要があります。既存の監査クラスのマスクを使用しないでください。

例 30–10 新しい監査クラスを作成する

この例では、監査イベントの小さなセットを保持するクラスを作成します。`audit_class` ファイルに追加されるエントリは次のとおりです。

```
0x10000000:pf:profile command
```

このエントリによって、`pf` という名前の新しい監査クラスが作成されます。例 30–11 で、この新しい監査クラスにデータを割り当てます。

注意事項 `audit_class` ファイルをカスタマイズした場合、`audit_user` に対する変更が新しい監査クラスと一致するようにします。`audit_user` 内の監査クラスが `audit_class` データベースのサブセットになっていないと、エラーが発生します。

▼ 監査イベントの所属先クラスの変更方法

既存の監査クラスのサイズを減らしたり、独自のクラスにイベントを入れたりするために、監査イベントのクラスメンバーシップを変更できます。1 つのシステム上で監査イベントから監査クラスへのマッピングを再構成する場合は、監査されているすべてのシステムに変更をコピーする必要があります。

1 **Primary Administrator 役割を引き受けるか、スーパーユーザーになります。**

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 **(省略可能) `audit_event` ファイルのバックアップコピーを保存します。**

```
# cp /etc/security/audit_event /etc/security/audit_event.orig
```

3 特定のイベントがどのクラスに属するかを変更するには、イベントの *class-list* を変更します。

各エントリの書式は次のとおりです。

number:*name*:*description*:*class-list*

number	監査イベント ID です。
name	監査イベントの名前です。
description	通常、監査レコードの作成を発生させるシステムコールまたは実行可能プログラム。
class-list	コンマ区切りの監査クラスの一覧です。

例 30–11 既存の監査イベントを新しいクラスにマッピングする

この例では、既存の監査イベントを例 30–10 で作成した新しいクラスにマッピングします。`audit_control` ファイルに、バイナリ監査レコードは、`pf` クラス内のイベントの成功または失敗を書き込みます。`syslog` 監査ログは、`pf` クラスのイベントの失敗だけを書き込みます。

```
# grep pf | /etc/security/audit_class
0x10000000:pf:profile command
# vi /etc/security/audit_event
6180:AUE_prof_cmd:profile command:ua,as,pf
# vi audit_control
...
flags:lo,pf
plugin:name=audit_binfile.so; p_dir=/var/audit; p_minfree=10
plugin:name=audit_syslog.so; p_flags=-lo,-pf
```

例 30–12 setuid プログラムの使用を監査する

この例では、setuid と setgid プログラムの呼び出しを監視するためにイベントを保持するクラスを作成します。バイナリ監査レコードは、lo クラスおよび na クラスのイベントの成功と失敗、st クラスのイベントの成功を書き込みます。syslog 監査ログは、st クラスのイベントの成功だけを書き込みます。

```
# vi /etc/security/audit_class
0x00000800:st:setuid class
# vi /etc/security/audit_event
26:AUE_SETGROUPS:setgroups(2):st
27:AUE_SETPGRP:setpgrp(2):st
40:AUE_SETREUID:setreuid(2):st
41:AUE_SETREGID:setregid(2):st
214:AUE_SETEGID:setegid(2):st
215:AUE_SETEUID:seteuid(2):st

# vi audit_control
## audit_control file
flags:lo,+st
naflags:lo,na
plugin:name=audit_binfile.so; p_dir=/var/audit; p_minfree=10
plugin:name=audit_syslog.so; p_flags=-lo,+st
```

監査サービスの構成と有効化 (作業マップ)

次の作業マップは、監査サービスの構成と有効化の手順を示しています。作業は順番に並んでいます。

作業	説明	参照先
1. (省略可能) 監査構成ファイルを変更します	監査を必要とするイベント、クラス、およびユーザーを選択します。	624 ページの「監査ファイルの構成 (作業マップ)」
2. 監査パーティションを作成します	監査ファイル用のディスク容量を作成し、ファイルのアクセス権を使用して監査ファイルを保護します。	635 ページの「監査ファイルのパーティションの作成方法」
3. audit_warn 別名を作成します	監査サービスに注意が必要なとき、電子メール警告を受信するユーザーを決定します。	639 ページの「audit_warn 電子メールエイリアスの構成方法」
4. (省略可能) 監査ポリシーを変更します	サイトに必要な追加の監査データを定義します。	639 ページの「監査ポリシーを構成する方法」
6. 非大域ゾーンの監査を構成します	非大域ゾーンで監査レコードが収集されるようにします	646 ページの「ゾーンでの監査サービスの構成 (手順)」

作業	説明	参照先
7. 監査を有効にします	監査サービスを有効にします。	642 ページの「監査サービスを有効にする方法」
	perzone 監査が有効になっている場合は、非大域ゾーンで監査を有効にします。	例 30–20
8. (省略可能) 監査を無効にします	監査サービスを無効にします。	644 ページの「監査サービスを無効にする方法」
	perzone 監査が有効になっている場合は、非大域ゾーンで監査を無効にします。	例 30–25
9. (省略可能) 監査構成変更を再読み込みします	auditd デーモンの実行中に監査構成の変更をカーネルに読み込みます。	645 ページの「監査サービスの更新方法」

監査サービスの構成と有効化 (手順)

構成ファイルをサイト用に構成したあと、監査ファイルのディスク容量を設定する必要があります。監査サービスのほかの属性を設定して、サービスを有効にする必要もあります。この節では、構成の設定を変更したときに監査サービスを更新する手順も説明します。

非大域ゾーンがインストールされている場合、監査対象の大域ゾーンと同じようにゾーンを監査することができます。非大域ゾーンを別々に監査する場合は、非大域ゾーンの監査構成ファイルを変更することができます。監査構成ファイルをカスタマイズする方法については、624 ページの「監査ファイルの構成 (作業マップ)」を参照してください。

▼ 監査ファイルのパーティションの作成方法

次の手順では、監査ファイル用のパーティションの作成方法、および監査に対応するファイルシステムとディレクトリの作成方法について説明します。すでに空のパーティションがある場合、またはすでに空のファイルシステムをマウントしている場合は、必要に応じて手順を省略してください。

1 **Primary Administrator 役割を引き受けるか、スーパーユーザーになります。**

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 必要なディスク容量を決定します。

ホストごとに 200M バイト以上を割り当てます。ただし、必要な監査の量によりディスク容量要件が決まります。つまり、要件によっては、この数値を超えることもあります。予備ディレクトリのローカルパーティション領域も含めてください。

3 必要に応じて、監査パーティションを作成します。

この手順は、サーバーのインストール時に実行するのがもっとも簡単です。サーバーにマウントされていないディスク上にパーティションを作成することもできます。パーティションの作成方法については、『Solaris のシステム管理 (デバイスとファイルシステム)』の第 11 章「ディスクの管理 (手順)」を参照してください。

```
# newfs /dev/rdsk/cwtxdysz
```

この例で、 /dev/rdsk/c*w*t *x*d*y*s*z* は、パーティションの raw デバイス名です。

ローカルホストを監査する場合は、ローカルホスト用に予備の監査ディレクトリも作成します。

4 新しいパーティションのマウント先を作成します。

```
# mkdir /var/audit/server-name.n
```

server-name.n は、サーバー名と各パーティションの識別番号を結合したものです。この識別番号は省略できますが、多数の監査ディレクトリを作成する場合はこの番号を使用すると便利です。

5 新しいパーティションを自動的にマウントするエントリを追加します。

/etc/vfstab ファイルに次のような行を追加します。

```
/dev/dsk/cwtxdysz /dev/rdsk/cwtxdysz /var/audit/server-name.n   ufs  2  yes
```

6 (省略可能) 各パーティションの最小空き容量のしきい値を削除します。

デフォルトの構成を使用した場合、ディレクトリの 80% がいっぱいになった時点で警告が生成されます。このため、パーティション上に空き容量を予約する必要はありません。

```
# tunefs -m 0 /var/audit/server-name.n
```

7 新しい監査パーティションをマウントします。

```
# mount /var/audit/server-name.n
```

8 新しいパーティションに監査ディレクトリを作成します。

```
# mkdir /var/audit/server-name.n/files
```

9 マウント先と新しいディレクトリへのアクセス権を訂正します。

```
# chmod -R 750 /var/audit/server-name.n/files
```

10 ファイルサーバー上で、ほかのホストからアクセスできるファイルシステムを定義します。

監査レコードを格納するために、ディスクファームをインストールすることがよく行われます。監査ディレクトリを複数のシステムで使用する場合は、そのディレクトリを NFS サービスを通して共有する必要があります。`/etc/dfs/dfstab` ファイルに対して、次のようなエントリをディレクトリごとに追加します。

```
share -F nfs /var/audit/server-name.n/files
```

11 ファイルサーバー上で、**NFS** サービスを起動し直します。

このコマンドが、ユーザーが起動した最初の 1 つまたは複数の `share` コマンドの場合、NFS デーモンが実行されていないことがあります。

- **NFS** サービスがオフラインの場合、サービスを有効にします。

```
% svcs \*nfs\*
disabled       Nov_02   svc:/network/nfs/rquota:default
offline        Nov_02   svc:/network/nfs/server:default
# svcadm enable network/nfs/server
```

- **NFS** サービスが実行中の場合、サービスを再起動します。

```
% svcs \*nfs\*
online         Nov_02   svc:/network/nfs/client:default
online         Nov_02   svc:/network/nfs/server:default
# svcadm restart network/nfs/server
```

NFS サービスの詳細は、『Solaris のシステム管理 (ネットワークサービス)』の「NFS サービスの設定」を参照してください。永続的なサービスの管理の詳細は、『Solaris のシステム管理 (基本編)』の第 18 章「サービスの管理 (概要)」および `smf(5)` のマニュアルページを参照してください。

例 30–13 予備の監査ディレクトリを作成する

監査サービスを実行するすべてのシステムには、利用できるファイルシステムがほかにない場合に使用するローカルファイルシステムが必要です。この例では、ファイルシステムが `egret` という名前のシステムに追加されます。このファイルシステムは、ローカルシステムだけで使用されるため、ファイルサーバーの手順は必要ありません。

```
# newfs /dev/rdsk/c0t2d0
# mkdir /var/audit/egret
# grep egret /etc/vfstab
/dev/dsk/c0t2d0s1  /dev/rdsk/c0t2d0s1  /var/audit/egret ufs  2  yes  -
# tunefs -m 0 /var/audit/egret
# mount /var/audit/egret
# mkdir /var/audit/egret/files
# chmod -R 750 /var/audit/egret/files
```

例 30–14 新しい監査パーティションを作成する

この例では、新しいファイルシステムが、2 つの新しいディスクに作成されます。この 2 つのディスクは、ネットワーク上のほかのシステムと共有します。

```
# newfs /dev/rdsk/c0t2d0
# newfs /dev/rdsk/c0t2d1
# mkdir /var/audit/egret.1
# mkdir /var/audit/egret.2
# grep egret /etc/vfstab
/dev/dsk/c0t2d0s1  /dev/rdsk/c0t2d0s1  /var/audit/egret.1 ufs  2  yes  -
/dev/dsk/c0t2d1s1  /dev/rdsk/c0t2d1s1  /var/audit/egret.2 ufs  2  yes  -
# tunefs -m 0 /var/audit/egret.1
# tunefs -m 0 /var/audit/egret.2
# mount /var/audit/egret.1
# mount /var/audit/egret.2
# mkdir /var/audit/egret.1/files
# mkdir /var/audit/egret.2/files
# chmod -R 750 /var/audit/egret.1/files /var/audit/egret.2/files
# grep egret /etc/dfs/dfstab
 share -F nfs /var/audit/egret.1/files
 share -F nfs /var/audit/egret.2/files
# svcadm enable network/nfs/server
```

例 30–15 ZFS 監査パーティションを作成する

この例では、管理者は、ZFS 監査パーティションの作成後に `script` コマンドを実行します。コマンドの出力は次のとおりです。

```
# zpool create auditf mirror c0t4d0 c0t5d0
# zfs create -o mountpoint=/audit auditf/audit
# zfs create auditf/audit/noddy
# zfs create auditf/audit/noddy/files
# zfs create auditf/audit/blinken
# zfs create auditf/audit/blinken/files
# zfs set devices=off auditf/audit
# zfs set exec=off auditf/audit
# zfs set setuid=off auditf/audit
# zfs set sharenfs=on auditf/audit
# share
-               /audit/blinken/files   rw   ""
-               /audit/noddy   rw   ""
-               /audit/blinken   rw   ""
-               /audit/noddy/files   rw   ""
-               /audit   rw   ""
# ^D
script done on Fri Apr 10 10:10:20 2009
```

次に、遠隔システム `remotesys` からのマウントを表示します。

```
# dfshares remotesys
RESOURCE                                SERVER ACCESS    TRANSPORT
remotesys:/audit/blinken/files       remotesys  -         -
remotesys:/audit/noddy               remotesys  -         -
remotesys:/audit/blinken             remotesys  -         -
```

```
remotesys:/audit/noddy/files         remotesys  -         -
remotesys:/audit                     remotesys  -         -
```

最後に、/audit ファイルシステムを /var/audit にマウントします。

```
# mount remotesys:/audit /var/audit
# ls /var/audit
blinken  noddy
```

▼ audit_warn 電子メールエイリアスの構成方法

audit_warn スクリプトは、audit_warn という電子メールエイリアスに対してメールを生成します。有効な電子メールアドレスにこのメールを送信するには、手順 2 で説明するオプションのいずれか 1 つに従います。

1 **Primary Administrator 役割を引き受けるか、スーパーユーザーになります。**

Primary Administrator 役割には、Primary Administrator プロファイルが含まれます。役割を作成してユーザーに役割を割り当てるには、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

2 **audit_warn 電子メールエイリアスを構成します。**

次のオプションのいずれかを選択します。

- オプション **1** – audit_warn スクリプトで、audit_warn 電子メールエイリアスをほかの電子メールアカウントに置き換えます。

  スクリプトの次の行で電子メールエイリアスを変更します。

  ```
  ADDRESS=audit_warn            # standard alias for audit alerts
  ```

- オプション **2** – audit_warn 電子メールをほかの電子メールアカウントにリダイレクトします。

  この場合、audit_warn 電子メールエイリアスを適切なメールエイリアスファイルに追加します。別名の追加先として、ローカルの /etc/mail/aliases ファイル、名前空間の mail_aliases データベースのいずれかを選択します。root 電子メールアカウントを audit_warn 電子メールエイリアスのメンバーとして登録する場合、新しいエントリは次のようになります。

  ```
  audit_warn: root
  ```

▼ 監査ポリシーを構成する方法

監査ポリシーを使用して、ローカルホストの監査レコードの特性を決定します。監査が有効な場合、/etc/security/audit_startup ファイルの内容により、監査ポリシーが決まります。

auditconfig コマンドを使用すると、現在の監査ポリシーオプションを検査および変更できます。また、監査ポリシーの変更内容を固定するために、audit_startup スクリプト内で auditconfig コマンドの監査ポリシーオプションを変更することも可能です。

1 **Audit Control プロファイルを含む役割を引き受けるか、スーパーユーザーになります。**

Audit Control プロファイルを含む役割を作成する方法およびユーザーに役割を割り当てる方法については、210 ページの「RBAC の構成 (作業マップ)」を参照してください。

2 監査ポリシーを確認します。

監査を有効にする前に、audit_startup ファイルの内容により、監査ポリシーが決まります。

```
#! /bin/sh
...
/usr/bin/echo "Starting BSM services."
/usr/sbin/auditconfig -setpolicy +cnt       Counts rather than drops records
/usr/sbin/auditconfig -conf       Configures event-class mappings
/usr/sbin/auditconfig -aconf      Configures nonattributable events
```

3 使用可能なポリシーオプションを表示します。

```
$ auditconfig -lspolicy
```

注 – perzone と ahlt ポリシーオプションは、大域ゾーンでのみ設定できます。

---

4 選択した監査ポリシーオプションを有効または無効にします。

```
# auditconfig -setpolicy prefixpolicy
```

prefix *prefix* 値 + を指定するとポリシーオプションが有効になります。*prefix* 値 - を指定するとポリシーオプションが無効になります。

policy 有効または無効にするポリシーを選択します。

このポリシーの設定は、次回ブートするまで、または auditconfig setpolicy コマンドを使ってポリシーを変更するまで持続します。

各ポリシーオプションの詳細は、617 ページの「監査ポリシーの決定」を参照してください。

例 30–16 cnt および ahlt 監査ポリシーオプションを設定する

この例では、cnt ポリシーが無効になり、ahlt ポリシーが有効になります。これらの設定では、監査パーティションがいっぱいで非同期イベントが発生した場合、シス

テムの使用が停止します。同期イベントが発生すると、スレッドを作成したプロセスがハングアップします。これらの設定は、セキュリティーが可用性よりも重要な場合に適しています。

次の audit_startup エントリによって、再起動を行なっても cnt ポリシーオプションが無効で ahlt ポリシーを有効のままになります。

```
# cat /etc/security/audit_startup
#!/bin/sh
/usr/bin/echo "Starting BSM services."
/usr/sbin/deallocate -Is
/usr/sbin/auditconfig -conf
/usr/sbin/auditconfig -aconf
/usr/sbin/auditconfig -setpolicy -cnt
/usr/sbin/auditconfig -setpolicy +ahlt
```

例 30–17 seq 監査ポリシーを一時的に設定する

この例では、auditd デーモンが実行中で、ahlt 監査ポリシーが設定されています。seq 監査ポリシーが現在のポリシーに追加されます。seq ポリシーは、sequence トークンをすべての監査レコードに追加します。監査レコードが破壊されたときやレコードが破棄されているとき、監査サービスのデバッグに役立ちます。

+ 接頭辞により、現在の監査ポリシーを seq に置き換えるのではなく、seq オプションが監査ポリシーに追加されます。auditconfig コマンドにより、ポリシーはコマンドが次に起動するまで、または次のブートまで有効になります。

```
$ auditconfig -setpolicy +seq
$ auditconfig -getpolicy
audit policies = ahlt,seq
```

例 30–18 perzone 監査ポリシーを設定する

この例では、perzone 監査ポリシーが、大域ゾーンの audit_startup スクリプト内で設定されます。ゾーンのブート時に、非大域ゾーンは、そのゾーン内の監査構成設定に関する監査レコードを収集します。

```
$ cat /etc/security/audit_startup
#!/bin/sh
/usr/bin/echo "Starting BSM services."
/usr/sbin/deallocate -Is
/usr/sbin/auditconfig -conf
/usr/sbin/auditconfig -aconf
/usr/sbin/auditconfig -setpolicy +perzone
/usr/sbin/auditconfig -setpolicy +cnt
```

例 30–19 監査ポリシーを変更する

この例では、auditd デーモンが実行中で、監査ポリシーが設定されています。auditconfig コマンドは、セッションの間、ahlt と cnt ポリシーを変更します。これらの設定により、監査ファイルシステムがいっぱいになると、監査レコードは破棄されますが、カウントされます。ahlt ポリシーの設定での制限については、手順 3 を参照してください。

```
$ auditconfig -setpolicy +cnt
$ auditconfig -setpolicy -ahlt
$ auditconfig -getpolicy
audit policies = cnt,seq
```

変更をaudit_startup ファイルに書き込むと、ポリシーは永続的に有効になります。

```
$ cat /etc/security/audit_startup
#!/bin/sh
/usr/bin/echo "Starting BSM services."
/usr/sbin/deallocate -Is
/usr/sbin/auditconfig -conf
/usr/sbin/auditconfig -aconf
/usr/sbin/auditconfig -setpolicy +cnt
```

ahlt ポリシーはデフォルトで無効になっているため、-ahlt オプションをファイル内で指定する必要はありません。この設定は、監査レコードのセキュリティーよりも可用性が重要な場合に適しています。

▼ 監査サービスを有効にする方法

この手順では、監査サービスをすべてのゾーンで有効にします。非大域ゾーンで監査デーモンを起動するには、例 30–20 を参照してください。

監査が安全に構成されると、監査が有効化されるまで、システムはシングルユーザーモードになります。マルチユーザーモードで監査を有効にすることもできます。

始める前に 次の作業を完了してから、この手順をスーパーユーザーとして実行してください。

- 計画 - 611 ページの「Oracle Solaris 監査の計画 (作業マップ)」
- 監査ファイルのカスタマイズ – 624 ページの「監査ファイルの構成 (作業マップ)」
- 監査パーティションの設定 – 635 ページの「監査ファイルのパーティションの作成方法」
- 監査警告メッセージの設定 – 639 ページの「audit_warn 電子メールエイリアスの構成方法」
- 監査ポリシーの設定 – 639 ページの「監査ポリシーを構成する方法」

注 – 監査が成功するには、ホスト名変換が正しく機能している必要があります。ネームサービスの hosts データベースが正しく構成され、機能している必要があります。

hosts データベースの構成については、nsswitch.conf(4) および netconfig(4) のマニュアルページを参照してください。詳細は、『Solaris のシステム管理 (ネーミングとディレクトリサービス : DNS、NIS、LDAP 編)』または『Solaris のシステム管理 (ネーミングとディレクトリサービス : NIS+ 編)』を参照してください。

1 監査サービスを有効にするスクリプトを実行します。

/etc/security ディレクトリに移動し、bsmconv スクリプトを実行します。

```
# cd /etc/security
# ./bsmconv
This script is used to enable the Basic Security Module (BSM).
Shall we continue with the conversion now? [y/n] y
bsmconv: INFO: checking startup file.
bsmconv: INFO: turning on audit module.
bsmconv: INFO: initializing device allocation.

The Basic Security Module is ready.
If there were any errors, please fix them now.
Configure BSM by editing files located in /etc/security.
Reboot this system now to come up with BSM enabled.
```

スクリプトの働きについては、bsmconv(1M) のマニュアルページを参照してください。

2 システムを再起動します。

```
# reboot
```

システムがマルチユーザーモードに移行すると、起動ファイル /etc/security/audit_startup によって auditd デーモンが自動的に動作します。

スクリプトのもう 1 つの働きは、デバイス割り当てを有効にすることです。デバイス割り当ての設定方法は、89 ページの「デバイス割り当ての管理 (作業マップ)」を参照してください。

例 30–20 非大域ゾーンで監査を有効にする

次の例では、監査が大域ゾーンで有効になり非大域ゾーンがブートされたあと、大域ゾーン管理者が perzone ポリシーを有効にします。非大域ゾーンのゾーン管理者は、ゾーンの監査ファイルを構成して、ゾーンで監査デーモンを起動します。

```
zone1# svcadm enable svc:/system/auditd
```

▼ 監査サービスを無効にする方法

ある時点で監査サービスが不要になった場合、この手順は監査が有効になる前のシステム状態にシステムを戻します。非大域ゾーンが監査されている場合は、その監査サービスも無効になります。

注意 – このコマンドによって、デバイス割り当ても無効になります。デバイス割り当てを可能にする場合は、このコマンドを実行しないでください。デバイス割り当てを保ったまま監査を無効にする方法については、例 30–21 を参照してください。

1 スーパーユーザーになり、システムをシングルユーザーモードにします。

```
% su
Password:       <Type root password>
# init S
```

詳細は、init(1M) のマニュアルページを参照してください。

2 スクリプトを実行して、監査を無効にします。

/etc/security ディレクトリに移動し、bsmunconv スクリプトを実行します。

```
# cd /etc/security
# ./bsmunconv
```

スクリプトのもう 1 つの働きは、デバイス割り当てを無効にすることです。

bsmunconv スクリプトの詳細は、bsmconv(1M) のマニュアルページを参照してください。

3 システムをマルチユーザーモードにします。

```
# init 6
```

例 30–21 デバイス割り当てを保持したまま監査を無効にする

この例では、監査サービスはレコードの収集を停止しますが、デバイス割り当ては引き続き機能します。audit_user ファイルのすべてのユーザーエントリと同様に、audit_control ファイルの flags、 naflags、 および plugin エントリからのすべての値が削除されます。

```
## audit_control file
flags:
naflags:

## audit_user file
```

auditd デーモンが実行されますが、監査レコードは保持されません。

例 30–22 監査をゾーン単位で無効にする

この例では、監査サービスが無効化された zone1 内で、監査サービスが実行を停止します。デバイス割り当ては引き続き機能します。perzone 監査ポリシーが設定されていない状態で、このコマンドが大域ゾーンで実行されると、大域ゾーンだけでなくすべてのゾーンで監査が無効になります。

```
zone1 # audit -t
```

▼ 監査サービスの更新方法

この手順では、デーモンの実行中に監査構成ファイルを変更した場合に、auditd デーモンを再起動します。

1. **Audit Control 権利プロファイルを含む役割を引き受けるか、あるいはスーパーユーザーになります。**

   Audit Control 権利プロファイルを含む役割の作成方法およびユーザーに役割を割り当てる方法については、210 ページの「RBAC の構成 (作業マップ)」を参照してください。

2. **適切なコマンドを選択します。**

   - **audit_control ファイルの naflags 行を変更する場合は、ユーザーに起因しないイベント用のカーネルマスクを変更します。**

     ```
     $ /usr/sbin/auditconfig -aconf
     ```

     リブートすることもできます。

   - **audit_control ファイルのほかの行を変更する場合は、audit_control ファイルを再度読み込みます。**

     監査デーモンは、audit_control ファイルからの情報を内部で格納します。新しい情報を使用するには、システムを再起動するか、または変更されたファイルを監査デーモンが読み込むようにします。

     ```
     $ /usr/sbin/audit -s
     ```

     注 – 監査レコードは、各プロセスに関連付けられた監査事前選択マスクに基づいて生成されます。audit -s を実行しても、既存のプロセス内のマスクは変更されません。既存プロセスの事前選択マスクを変更するには、プロセスを再起動する必要があります。リブートすることもできます。

     ---

     audit -s コマンドを使用すると、監査デーモンはディレクトリと最小限の空きの値を audit_control ファイルから再度読み取ります。このコマンドにより、後続のログインにより発生するプロセス用の事前選択マスクの生成が変更されます。

- 監査デーモンの実行中に **audit_event** ファイルまたは **audit_class** ファイルを変更する場合は、監査サービスを再表示します。

  変更したイベントからクラスへのマッピングをシステムに読み込み、マシンを使用する各ユーザーが正しく監査されているかを確認します。

  ```
  $ auditconfig -conf
  $ auditconfig -setumask auid classes
  ```

  auid ユーザー ID です。

  classes 事前に選択された監査クラスです。

  例については、669 ページの「ユーザーの事前選択マスクを変更する方法」を参照してください。

- 稼動中のシステムで監査ポリシーを変更するには、例 **30–17** を参照してください。

例 **30–23** 監査デーモンを再起動する

この例では、システムをシングルユーザーモードにしたあとで、マルチユーザーモードに戻します。システムがマルチユーザーモードになったとき、変更された構成ファイルがシステムに読み込まれます。

```
# init S
# init 6
```

ゾーンでの監査サービスの構成 (手順)

監査サービスは、ゾーン内での監査イベントも含め、システム全体を監査します。非大域ゾーンがインストールされたシステムでは、すべてのゾーンを同様に監査することも、ゾーンごとに監査を制御することもできます。背景情報については、607 ページの「Oracle Solaris ゾーンを含むシステムでの監査」を参照してください。計画を立てるには、612 ページの「ゾーン内の監査の計画方法」を参照してください。

▼ すべてのゾーンの監査を同様に構成する方法

この手順に従えば、すべてのゾーンを同様に監査できます。この方法を使えば、必要とされるコンピュータオーバーヘッドと管理リソースが最小になります。

1 大域ゾーンの監査を構成します。

a. **624** ページの「監査ファイルの構成 (作業マップ)」の作業を行います。

b. 634 ページの「監査サービスの構成と有効化 (作業マップ)」の作業を行います。ただし、次の点は例外になります。

- perzone 監査ポリシーを有効にしないでください。
- 監査サービスを有効にしないでください。監査サービスの有効化は、非大域ゾーンの監査の構成が完了したあとで行います。

2 大域ゾーンからすべての非大域ゾーンに、監査構成ファイルをコピーします。

編集した次のファイルをすべてコピーします。 audit_class、 audit_control、 audit_event、 audit_user。audit_startup と audit_warn はコピーしないでください。編集していないファイルをコピーする必要はありません。

2 つの選択肢があります。スーパーユーザーとして、ファイルをコピーすることも、ファイルをループバックマウントすることもできます。非大域ゾーンが動作している必要があります。

- ファイルをコピーします。

  a. 大域ゾーンから、非大域ゾーンの **/etc/security** ディレクトリを一覧表示します。

  ```
  # ls /zone/zonename/etc/security/
  ```

  b. ゾーンの **/etc/security** ディレクトリに監査構成ファイルをコピーします。

  ```
  # cp /etc/security/audit-file /zone/zonename/etc/security/audit-file
  ```

  その後、ある監査構成ファイルを大域ゾーンで変更した場合には、そのファイルを非大域ゾーンにコピーし直します。

- 構成ファイルをループバックマウントします。

  a. 大域ゾーンから、非大域ゾーンを停止します。

  ```
  # zoneadm -z non-global-zone halt
  ```

  b. 大域ゾーンで変更した監査構成ファイルごとに読み取り専用のループバックマウントを **1** つずつ作成します。

  ```
  # zonecfg -z non-global-zone
   add fs
      set special=/etc/security/audit-file
      set dir=/etc/security/audit-file
      set type=lofs
      add options [ro,nodevices,nosetuid]
      end
   exit
  ```

  c. 変更を有効にするには、非大域ゾーンをブートします。

  ```
  # zoneadm -z non-global-zone boot
  ```

システムをリブートしても構いません。

その後、ある監査構成ファイルを大域ゾーンで変更した場合には、非大域ゾーンにループバックマウントされたファイルを更新するためにシステムをリブートします。

例 30–24 監査構成ファイルをループバックマウントする

この例では、システム管理者が audit_class、 audit_event、 audit_control、 audit_user、 audit_startup、 および audit_warn ファイルを変更しました。

audit_startup および audit_warn ファイルは、大域ゾーンでしか読み取られないため、非大域ゾーンでループバックマウントする必要はありません。

このシステム machine1 上で、管理者は 2 つの非大域ゾーン machine1–webserver と machine1–appserver を作成しました。管理者が監査構成ファイルのカスタマイズを完了しました。管理者があとでファイルを変更した場合は、変更を有効にするためにシステムがリブートされます。

```
# zoneadm -z machine1-webserver halt
# zoneadm -z machine1-appserver halt
# zonecfg -z machine1-webserver
 add fs
    set special=/etc/security/audit_class
    set dir=/etc/security/audit_class
    set type=lofs
    add options [ro,nodevices,nosetuid]
    end
 add fs
    set special=/etc/security/audit_event
    set dir=/etc/security/audit_event
    set type=lofs
    add options [ro,nodevices,nosetuid]
    end
 add fs
    set special=/etc/security/audit_control
    set dir=/etc/security/audit_control
    set type=lofs
    add options [ro,nodevices,nosetuid]
    end
add fs
    set special=/etc/security/audit_user
    set dir=/etc/security/audit_user
    set type=lofs
    add options [ro,nodevices,nosetuid]
    end
 exit
# zonecfg -z machine1-appserver
 add fs
    set special=/etc/security/audit_class
    set dir=/etc/security/audit_class
    set type=lofs
    add options [ro,nodevices,nosetuid]
```

```
    end
...
 exit
```

ゾーンがリブートされると、監査構成ファイルはゾーン内で読み取り専用になります。

▼ ゾーンごとの監査を構成する方法

この手順に従えば、個々のゾーン管理者が自身のゾーン内で監査サービスを制御できます。ポリシーオプションの完全な一覧については、`auditconfig(1M)` のマニュアルページを参照してください。

1. 大域ゾーンで監査を構成します。ただし、監査サービスは有効にしないでください。

   a. 624 ページの「監査ファイルの構成 (作業マップ)」の作業を行います。

   b. 634 ページの「監査サービスの構成と有効化 (作業マップ)」の作業を行います。ただし、次の点は例外になります。

      - `perzone` 監査ポリシーを追加してください。例については、例 30–18 を参照してください。
      - 監査サービスを有効にしないでください。監査サービスの有効化は、非大域ゾーンの監査の構成が完了したあとで行います。

2. 各非大域ゾーンで監査ファイルを構成します。

   注 – 監査を無効にする予定の非大域ゾーンでは、この手順をスキップできます。監査を無効にするには、例 30–25 を参照してください。

   ---

   a. 624 ページの「監査ファイルの構成 (作業マップ)」の作業を行います。

   b. 634 ページの「監査サービスの構成と有効化 (作業マップ)」で説明した手順に従います。

   c. システム全体の監査設定は構成しないでください。

      具体的には、非大域ゾーンの `audit_startup` ファイルに `perzone` や `ahlt` ポリシーを追加しないでください。また、非大域ゾーンから `bsmconv` コマンドを実行しないでください。

d. 使用するゾーンで監査を有効にします。

監査の構成後に大域ゾーンをリブートすると、使用するゾーンの監査が自動的に有効になります。

システムのリブート後に大域ゾーン管理者が perzone 監査ポリシーを有効にした場合、個々のゾーン管理者が監査の有効化を行う必要があります。詳細は、例 30–20 を参照してください。

3 大域ゾーンで監査サービスを有効にします。

手順については、642 ページの「監査サービスを有効にする方法」を参照してください。

例 30–25 非大域ゾーンで監査を無効にする

この例は、大域ゾーンで perzone 監査ポリシーが設定されている場合に正しく機能します。noaudit ゾーンのゾーン管理者が、そのゾーンの監査を無効にします。この管理者は、監査を無効にするつもりであったため、監査構成ファイルを編集していませんでした。

```
noauditzone # svcadm disable svc:/system/auditd
```

監査レコードの管理 (作業マップ)

次の作業マップは、監査レコードの選択、分析、および管理の手順を示しています。

作業	説明	参照先
監査レコードの書式を表示します	監査イベント用に収集される情報、および表示される情報の順番を表示します。	651 ページの「監査レコードの書式の表示方法」
監査レコードをマージします	複数のマシンの監査ファイルを 1 つの監査トレールに結合します。	652 ページの「監査トレールの監査ファイルをマージする方法」
調査するレコードを選択します	調査対象の特定のイベントを選択します。	655 ページの「監査トレールから監査イベントを選択する方法」
監査レコードを表示します	バイナリ監査レコードの表示を有効にします。	657 ページの「バイナリ監査ファイルの内容を表示する方法」
正確でない名前を付けられた監査ファイルを整理します	監査サービスにより意図的でなく開いたままにされた監査ファイルに最終タイムスタンプを設定します。	659 ページの「not_terminated 監査ファイルを整理する方法」

作業	説明	参照先
監査トレールのオーバーフローを防止します	監査ファイルシステムがいっぱいになることを防止します。	660 ページの「監査トレールのオーバーフローを防ぐ方法」

監査レコードの管理

監査トレールを管理することによって、ネットワーク上のユーザーの動作を監視することができます。監査プロセスを行うと、大量のデータが生成される可能性があります。次の手順では、さまざまな監査データを使用して作業を行う方法について説明します。

▼ 監査レコードの書式の表示方法

必要な監査データを検索するスクリプトを作成するためには、監査イベント内のトークンの順番を知る必要があります。bsmrecord コマンドは、監査イベントの監査イベント番号、監査クラス、選択マスク、およびレコード書式を表示します。

- すべての監査イベントレコードを **HTML** ファイルに出力します。

  -a オプションを指定すると、すべての監査イベントレコードの書式が表示されます。-h オプションを指定すると、ブラウザで表示できる HTML 形式で一覧が出力されます。

  ```
  % bsmrecord -a -h > audit.events.html
  ```

  ブラウザで *html ファイルを表示する場合は、ブラウザの検索ツールを使用して特定のレコードを検索します。

  詳細は、bsmrecord(1M) のマニュアルページを参照してください。

例 30–26 プログラムの監査レコード書式を表示する

この例では、login プログラムによって生成されたすべての監査レコードの書式を表示します。login プログラムには、rlogin、 telnet、 newgrp、 Solaris 管理コンソールへの役割ログイン、および Oracle Solaris Secure Shell も含まれます。

```
% bsmrecord -p login
login: logout
  program     various              See login(1)
  event ID    6153                 AUE_logout
...

newgrp
  program     newgrp               See newgrp login
  event ID    6212                 AUE_newgrp_login
...
```

```
rlogin
  program     /usr/sbin/login      See login(1) - rlogin
  event ID    6155                 AUE_rlogin
...

SMC: role login
  program     SMC server           See role login
  event ID    6173                 AUE_role_login
...

/usr/lib/ssh/sshd
  program     /usr/lib/ssh/sshd    See login - ssh
  event ID    6172                 AUE_ssh
...

telnet login
  program     /usr/sbin/login      See login(1) - telnet
  event ID    6154                 AUE_telnet
  ...
```

例 30–27 ある監査クラスの監査レコード書式を表示する

この例では、fd クラスに属するすべての監査レコードの書式を表示します。

```
% bsmrecord -c fd

rmdir
  system call rmdir                See rmdir(2)
  event ID    48                   AUE_RMDIR
  class       fd                   (0x00000020)
      header
      path
      [attribute]
      subject
      [use_of_privilege]
      return

unlink
  system call unlink               See unlink(2)
  event ID    6                    AUE_UNLINK
  ...

unlinkat
  system call unlinkat             See openat(2)
  event ID    286                  AUE_UNLINKAT
  ...
```

▼ 監査トレールの監査ファイルをマージする方法

すべての監査ディレクトリ内のすべての監査ファイルをマージすると、監査トレール全体の内容を分析できます。auditreduce コマンドを使用すると、入力ファイルのすべてのレコードが 1 つの出力ファイルにマージされます。マージが完了すると、入力ファイルを削除できます。出力ファイルが

/etc/security/audit/*server-name*/files という名前のディレクトリに配置されている場合、完全パスを指定しなくても、auditreduce コマンドは出力ファイルを検索できます。

注 – この手順は、バイナリ監査レコードだけに適用します。

---

1 **Audit Review** プロファイルを含む役割を引き受けるか、スーパーユーザーになります。

System Administrator 役割には、 Audit Review プロファイルが含まれます。Audit Review プロファイルを含む役割を別々に作成することもできます。役割を作成する方法と役割をユーザーに割り当てる方法については、210 ページの「RBAC の構成 (作業マップ)」を参照してください。

2 マージされた監査ファイルを格納するディレクトリを作成します。

```
# mkdir audit-trail-directory
```

3 ディレクトリへのアクセスを制限します。

```
# chmod 700 audit-trail-directory
# ls -la audit-trail-directory
drwx------   3 root     sys          512 May 12 11:47 .
drwxr-xr-x   4 root     sys         1024 May 12 12:47 ..
```

4 監査トレール内の監査レコードをマージします。

ディレクトリを *audit-trail-directory* に変更して、指定した接尾辞を持つファイルに監査レコードをマージします。ローカルシステム上にある audit_control ファイルの dir 行に指定されているすべてのディレクトリがマージされます。

```
# cd audit-trail-directory
# auditreduce -Uppercase-option -O suffix
```

大文字オプションを auditreduce コマンドに指定すると、監査トレール内のファイルを操作できます。次の大文字オプションがあります。

-A 監査トレール内のすべてのファイルを選択します。

-C 完全ファイルだけを選択します。このオプションは、接尾辞 not_terminated を持つファイルを無視します。

-M 特定の接尾辞を持つファイルを選択します。接尾辞はマシン名、またはサマリーファイルに指定した接尾辞です。

-O 開始時刻と終了時刻を示す 14 文字のタイムスタンプおよび接尾辞 *suffix* が付いた監査ファイルを現在のディレクトリに作成します。

例 30–28 サマリーファイルに監査ファイルをコピーする

次の例では、System Administrator 役割 sysadmin は、すべてのファイルを監査トレールからマージされたファイルにコピーします。

```
$ whoami
sysadmin
$ mkdir /var/audit/audit_summary.dir
$ chmod 700 /var/audit/audit_summary.dir
$ cd /var/audit/audit_summary.dir
$ auditreduce -A -O All
$ ls *All
20100827183214.20100827215318.All
```

次の例では、完全ファイルだけが監査トレールからマージされたファイルにコピーされます。

```
$ cd /var/audit/audit_summary.dir
$ auditreduce -C -O Complete
$ ls *Complete
20100827183214.20100827214217.Complete
```

次の例では、完全ファイルだけが example1 マシンからマージされたファイルにコピーされます。

```
$ cd /var/audit/audit_summary.dir
$ auditreduce -M example1 -O example1summ
$ ls *summ
20100827183214.20100827214217.example1summ
```

例 30–29 監査ファイルをサマリーファイルに移動する

auditreduce で - D オプションを指定すると、監査ファイルをほかの場所にコピーしたときにその監査ファイルを削除します。次の例では、あるシステムの完全監査ファイルを、あとで調べるためにサマリーディレクトリにコピーします。

```
$ cd /var/audit/audit_summary.dir
$ auditreduce -C -O daily_example1 -D example1
$ ls *example1
20100827183214.20100827214217.daily_example1
```

このコマンドが正常に完了すると、*daily_example1 ファイルへの入力となった example1 システムの監査ファイルが削除されます。

▼ 監査トレールから監査イベントを選択する方法

調べる監査レコードをフィルタリングすることができます。フィルタリングオプションの一覧については、auditreduce(1M) のマニュアルページを参照してください。

1. **Audit Review** プロファイルを含む役割を引き受けるか、スーパーユーザーになります。

   System Administrator 役割には、 Audit Review プロファイルが含まれます。Audit Review プロファイルを含む役割を別々に作成することもできます。役割を作成する方法と役割をユーザーに割り当てる方法については、210 ページの「RBAC の構成 (作業マップ)」を参照してください。

2. 監査トレールまたは指定した監査ファイルから必要なレコードを選択します。

   `auditreduce` *-lowercase-option argument* [*optional-file*]

argument	小文字オプションが必要とする特定の引数。たとえば、`-c` オプションは、`ua` などの監査クラスの *argument* を必要とします。
`-d`	特定の日付のイベントをすべて選択します。*argument* の日付の形式は、*yyyymmdd* です。ほかの日付オプション `-b` と `-a` は、特定の日付の前と後のイベントを選択します。
`-u`	特定のユーザーのイベント属性をすべて選択します。*argument* はユーザー名です。もう 1 つのユーザーオプション `-e` は、実効ユーザー ID のイベント属性をすべて選択します。
`-c`	事前選択された監査クラス内のイベントをすべて選択します。*argument* は監査クラス名です。
`-m`	特定の監査イベントのインスタンスをすべて選択します。*argument* は監査イベントです。
optional-file	監査ファイルの名前です。

例 30–30 監査ファイルを結合して削減する

`auditreduce` コマンドを使用すると、入力ファイルの結合時に不要なレコードを除外できます。たとえば、`auditreduce` コマンドを使用して、1 か月以上経過した監査ファイルから、ログインレコードとログアウトレコード以外のレコードを削除することができます。監査トレール全体が必要になった場合は、バックアップメディアから監査トレールを復元します。

```
# cd /var/audit/audit_summary.dir
# auditreduce -O lo.summary -b 20100827 -c lo; compress *lo.summary
```

例 30–31 na 監査レコードをサマリーファイルにコピーする

この例では、監査トレール内の、ユーザーに起因しない監査イベントのすべてのレコードが、1 つのファイルに収集されます。

```
$ whoami
sysadmin
$ cd /var/audit/audit_summary.dir
$ auditreduce -c na -O nasumm
$ ls *nasumm
20100827183214.20100827215318.nasumm
```

マージされた nasumm 監査ファイルは、na レコードの開始時刻と終了時刻のタイムスタンプが記録されます。

例 30–32 指定した監査ファイル内で監査イベントを検索する

監査ファイルを手動で選択して、指定された一連のファイルだけを検索できます。たとえば、前の例の *nasumm ファイルをさらに処理して、システムブートイベントを検索できます。これを行うには、auditreduce コマンドの最後の引数にファイル名を指定します。

```
$ auditreduce -m 113 -O systemboot 20100827183214.20100827215318.nasumm
20100827183214.20100827183214.systemboot
```

20100827183214.20100827183214.systemboot ファイルは、システムブート監査イベントだけを含みます。

例 30–33 ユーザー監査レコードをサマリーファイルにコピーする

この例では、特定のユーザー名を含む監査トレール内のレコードがマージされます。-e オプションを指定すると実効ユーザーが検索されます。-u オプションを指定すると監査ユーザーが検索されます。

```
$ cd /var/audit/audit_summary.dir
$ auditreduce -e tamiko -O tamiko
```

このファイル内の特定のイベントを検索できます。次の例では、2010 年 9 月 7 日にユーザーがログインした時間が確認されます。接尾辞にユーザーの名前が付いたファイルだけが確認されます。日付の短い形式は、*yyyymmdd* です。

```
# auditreduce -M tamiko -O tamikolo -d 20100907 -u tamiko -c lo
```

例 30–34 選択レコードを 1 つのファイルにコピーする

この例では、特定の日付におけるログイン、ログアウトのメッセージが監査トレールから選択されます。これらのメッセージは対象ファイルにマージされます。対象ファイルは、通常の監査ルートディレクトリ以外のディレクトリに書き込まれます。

```
# auditreduce -c lo -d 20100827 -O /var/audit/audit_summary.dir/logins
# ls /var/audit/audit_summary.dir/*logins
/var/audit/audit_summary.dir/20100827183936.20100827232326.logins
```

▼ バイナリ監査ファイルの内容を表示する方法

praudit コマンドを使用すると、バイナリ監査ファイルの内容を表示できます。auditreduce コマンドから出力をパイプしたり、特定の監査ファイルを読み取ったりできます。さらに処理する場合に -x オプションが役立ちます。

1. **Audit Review プロファイルを含む役割を引き受けるか、スーパーユーザーになります。**

   System Administrator 役割には、 Audit Review プロファイルが含まれます。Audit Review プロファイルを含む役割を別々に作成することもできます。役割を作成する方法と役割をユーザーに割り当てる方法については、210 ページの「RBAC の構成 (作業マップ)」を参照してください。

2. **次の praudit コマンドの 1 つを使用して、目的に沿った出力を生成します。**

   次の例は、同じ監査イベントからの praudit 出力を表示します。監査ポリシーは、sequence トークンと trailer トークンを含むように設定されています。

   - praudit -s コマンドを使用して、短い書式で 1 行につき 1 トークンの監査レコードを表示します。-l オプションを指定して、1 行に各レコードを配置します。

     ```
     $ auditreduce -c lo | praudit -s
     header,101,2,AUE_rlogin,,example1,2010-10-13 11:23:31.050 -07:00
     subject,jdoe,jdoe,staff,jdoe,staff,749,749,195 1234 server1
     text,successful login
     return,success,0
     sequence,1298
     ```

   - praudit -r コマンドを使用して、監査レコードの raw 書式で 1 行につき 1 トークンの監査レコードを表示します。-l オプションを指定して、1 行に各レコードを配置します。

     ```
     $ auditreduce -c lo | praudit -r
     21,101,2,6155,0x0000,192.168.60.83,1062021202,64408258
     36,2026700,2026700,10,2026700,10,749,749,195 1234 192.168.60.17
     40,successful login
     ```

```
39,0,0
47,1298
```

- praudit -x コマンドを使用して、XML 形式で 1 行につき 1 トークンの監査レコードを表示します。-l オプションを指定して、1 レコードの XML 出力を 1 行に配置します。

```
$ auditreduce -c lo | praudit -x
<record version="2" event="login - rlogin" host="example1"
time="Wed Aug 27 14:53:22 PDT 2010" msec="64">
<subject audit-uid="jdoe" uid="jdoe" gid="staff" ruid="jdoe"
rgid="staff" pid="749" sid="749" tid="195 1234 server1"/>
<text>successful login</text>
<return errval="success" retval="0"/>
<sequence seq-num="1298"/>

</record>
```

例 30–35 監査トレール全体を印刷する

lp コマンドにパイプすると、監査トレール全体の出力がプリンタに送られます。プリンタへのアクセスを制限してください。

```
# auditreduce | praudit | lp -d example.protected.printer
```

例 30–36 特定の監査ファイルを表示する

この例では、サマリーログインファイルを端末ウィンドウで調べます。

```
# cd /var/audit/audit_summary.dir/logins
# praudit 20100827183936.20100827232326.logins | more
```

例 30–37 監査レコードを XML 形式に変換する

この例では、監査レコードを XML 形式に変換します。

```
# praudit -x 20100827183214.20100827215318.logins > 20100827.logins.xml
```

*xml ファイルはブラウザを使って表示できます。スクリプトを使えば、XML ファイルの内容を操作して目的の情報を抽出できます。

注意事項 次のようなメッセージは、praudit コマンドを使用するために必要な権限がないことを示しています。

```
praudit: Can’t assign 20090408164827.20090408171614.example1 to stdin.
```

▼ not_terminated 監査ファイルを整理する方法

監査ファイルが開いている状態で監査デーモンが終了することがあります。また、サーバーがアクセス不能になって、強制的に別のサーバーに切り替わってしまうことがあります。このような場合、その監査ファイルは監査レコードとして使用されなくなりますが、監査ファイルの終了タイムスタンプとして文字列 not_terminated が付いたままになります。auditreduce -O コマンドを使用して、ファイルに正しいタイムスタンプを付けます。

1. 監査ファイル上の **not_terminated** 文字列が付いたファイルを作成順に一覧表示します。

   ```
   # ls -R1t audit-directory*/files/* | grep not_terminated
   ```

   -R　サブディレクトリ内のファイルを一覧表示します。

   -t　最新のファイルからもっとも古いファイルの順で一覧表示します。

   -1　1 列でファイルを一覧表示します。

2. 古い **not_terminated** ファイルを整理します。

   古いファイルの名前を auditreduce -O コマンドに指定します。

   ```
   # auditreduce -O system-name old-not-terminated-file
   ```

3. 古い **not_terminated** ファイルを削除します。

   ```
   # rm system-name old-not-terminated-file
   ```

例 30–38　閉じた not_terminated 監査ファイルを整理する

次の例では、not_terminated ファイルを検索し、名前を変更して、元のファイルを削除します。

```
ls -R1t */files/* | grep not_terminated
.../egret.1/20100908162220.not_terminated.egret
.../egret.1/20100827215359.not_terminated.egret
# cd */files/egret.1
# auditreduce -O egret 20100908162220.not_terminated.egret
# ls -1t
20100908162220.not_terminated.egret      Current audit file
20100827230920.20100830000909.egret      Input (old) audit file
20100827215359.not_terminated.egret
# rm 20100827215359.not_terminated.egret
# ls -1t
20100908162220.not_terminated.egret      Current audit file
20100827230920.20100830000909.egret      Cleaned up audit file
```

新しいファイルの開始タイムスタンプは、not_terminated ファイル内にある最初の監査イベントの時間を反映します。終了タイムスタンプは、そのファイル内にある最後の監査ファイルの時間を反映します。

▼ 監査トレールのオーバーフローを防ぐ方法

セキュリティーポリシーの関係ですべての監査データを保存する必要がある場合は、次の手順に従います。

1. 監査ファイルを定期的に保存するスケジュールを設定します。
   ファイルをオフラインのメディアにバックアップして、監査ファイルを保管します。これらのファイルを保存ファイルシステムに移動することもできます。

   syslog ユーティリティーを使用してテキスト監査ログを収集している場合、テキストログを保管します。詳細は、logadm(1M) のマニュアルページを参照してください。

2. スケジュールを設定して、保管された監査ファイルを監査ファイルシステムから削除します。

3. 補助情報を保存し保管します。
   監査レコードの解釈に必要な情報を、監査トレールとともに格納します。

4. 保管した監査ファイルの記録をとります。

5. 保存したメディアを適切な方法で保管します。

6. サマリーファイルを作成して、格納する監査データのボリュームを削減します。
   監査トレールからサマリーファイルを抽出するには、auditreduce コマンドのオプションを使用します。サマリーファイルには、指定された種類の監査イベントのレコードだけが含まれます。サマリーファイルを抽出するには、例 30–30 および例 30–34 を参照してください。

Oracle Solaris 監査の障害追跡 (手順)

この節では、さまざまな Oracle Solaris 監査のエラーメッセージ、設定、ほかのツールによる監査について説明します。ここで示す手順は、使用中のシステムで必要な監査イベントを記録する際に役立ちます。

Oracle Solaris 監査の障害追跡 (作業マップ)

次の作業マップでは、Oracle Solaris 監査のトラブルシューティングの手順を示します。

問題	解決方法	参照先
監査を構成したときに監査ファイルが作成されないのはなぜでしょうか。	監査デーモンと監査構成ファイルのトラブルシューティングを行います。	661 ページの「Oracle Solaris 監査が実行中であるかどうかを判定する方法」
収集される監査情報を減らすには、どうすればよいでしょうか。	監査が必要なイベントについてのみ、監査を行います。	664 ページの「生成される監査レコードの量を削減する方法」
システムでのユーザーの動作をすべて監査するには、どうすればよいでしょうか。	すべてのコマンドについて 1 人または複数のユーザーを監査します。	666 ページの「ユーザーによるすべてのコマンドを監査する方法」
記録される監査イベントを変更して、その変更内容を既存のセッションに適用するには、どうすればよいでしょうか。	ユーザーの事前選択マスクを更新します。	669 ページの「ユーザーの事前選択マスクを変更する方法」
変更内容を特定のファイルに格納するには、どうすればよいでしょうか。	ファイルの変更を監査し、`auditreduce` コマンドを使用して特定のファイルを検索します。	668 ページの「特定のファイルに対する変更の監査レコードを検索する方法」
監査ファイルのサイズを減らすには、どうすればよいでしょうか。	バイナリ監査ファイルのサイズを制限します。	671 ページの「バイナリ監査ファイルのサイズを制限する方法」
`audit_event` ファイルから監査イベントを削除するには、どうすればよいでしょうか。	`audit_event` ファイルを更新します。	670 ページの「特定のイベントが監査されないようにする方法」
Oracle Solaris システムへのすべてのログインを監査するには、どうすればよいでしょうか。	システムからのログインを監査します。	671 ページの「ほかの OS からのログインを監査する方法」
FTP 転送で監査レコードが保持されないのはなぜでしょうか。	ログを生成するユーティリティーに適切な監査ツールを使用します。	672 ページの「FTP および SFTP ファイル転送を監査する方法」

▼ Oracle Solaris 監査が実行中であるかどうかを判定する方法

監査が有効になっているはずだが、1 次監査ディレクトリに監査レコードがない場合、次の手順を試してください。

始める前に　ネームサービスの hosts データベースが正しく構成され、機能しています。ネームサービスの問題をデバッグする場合は、次の情報を参照してください。

- nsswitch.conf(4) のマニュアルページ
- 『Solaris のシステム管理 (ネーミングとディレクトリサービス : DNS、NIS、LDAP 編)』
- 『Solaris のシステム管理 (ネーミングとディレクトリサービス : NIS+ 編)』

1 監査が実行中であるかどうかを判定します。

- **c2audit** カーネルモジュールがロード済みであることを確認します。

  ```
  # modinfo | grep c2audit
  ```

  リストがない場合、監査が実行中でないことを示しています。次のリストは、監査が実行中であることを示しています。

  ```
  40  132ce90  14230 186   1  c2audit (C2 system call)
  ```

- 監査デーモンが動作していることを確認します。

  auditd サービスの状態を確認します。次のリストは、監査が実行中でないことを示しています。

  ```
  # svcs -x auditd
  svc:/system/auditd:default (Solaris audit daemon)
   State: disabled since Fri Aug 14 19:02:35 2009
  Reason: Disabled by an administrator.
     See: http://sun.com/msg/SMF-8000-05
     See: auditd(1M)
     See: audit(1M)
  Impact: This service is not running.
  ```

  次のリストは、監査サービスが実行中であることを示しています。

  ```
  # svcs auditd
  STATE          STIME    FMRI
  online         10:10:10 svc:/system/auditd:default
  ```

- 現在の監査の状況を確認します。

  次のリストは、監査が実行中でないことを示しています。

  ```
  # auditconfig -getcond
  auditconfig: auditon(2) failed.
  auditconfig: error = Operation not supported(48)
  ```

  次のリストは、監査が実行中であることを示しています。

  ```
  # auditconfig -getcond
  audit condition = auditing
  ```

監査サービスが実行中でない場合、有効にします。手順については、642 ページの「監査サービスを有効にする方法」を参照してください。

2 **audit_control** ファイルの構文を確認します。

```
# audit -v /etc/security/audit_control
audit: audit_control must have either a valid "dir:" entry
or a valid "plugin:" entry with "p_dir:" specified.
```

エラーを修正します。syntax ok というメッセージは、ファイルの構文が正しいことを示しています。

3 **audit_control** ファイルについて、**flags** キーワードおよび **naflags** キーワードの値が有効であることを確認します。

```
# grep flags /etc/security/audit_control
flags:lo
naflags:na,lp
```

audit_control ファイルに無効な値が含まれている場合、有効な値を指定します。前述の例で、lp は無効なクラスです。

4 **audit_user** ファイルについて、すべてのユーザーの値が有効であることを確認します。

```
# tail audit_user
...
# User Level Audit User File
#
# File Format
#
#    username:always:never
#
root:lo:no
admin:lp:no
```

audit_user ファイルに無効な値が含まれている場合、有効な値を指定します。前述の例で、lp は無効なクラスです。

5 カスタマイズ監査クラスを作成した場合、そのクラスにイベントが割り当て済みであることを確認します。

たとえば、次の audit_control ファイルには、Oracle Solaris ソフトウェアが配信していないクラスが含まれています。

```
# grep flags /etc/security/audit_control
flags:lo,pf
naflags:na,lo
```

pf クラスの作成については、632 ページの「監査クラスの追加方法」を参照してください。

a. クラスが **audit_class** ファイルで定義されていることを確認します。

監査クラスマスクは一意である必要があります。

```
# grep pf /etc/security/audit_class
0x10000000:pf:profile command
```

クラスが定義されていない場合、定義します。そうでない場合、audit_control ファイルおよび audit_user ファイルからクラスを削除します。

b. イベントがクラスに割り当てられていることを確認します。

```
# grep pf /etc/security/audit_event
6180:AUE_prof_cmd:profile command:ua,as,pf
```

イベントがクラスに割り当てられていない場合、適切なイベントをこのクラスに割り当てます。

6 前述の手順で問題が見つからなかった場合、システムログファイル **/var/adm/messages** および **/var/log/syslog** を確認します。

a. 問題を検出して修正します。

b. 次に、監査サービスが実行中である場合、再起動します。

```
# audit -s
```

c. 監査サービスが実行中でない場合、有効にします。
手順については、642 ページの「監査サービスを有効にする方法」を参照してください。

▼ 生成される監査レコードの量を削減する方法

使用しているシステムで監査する必要のあるイベントを決定した後、次に示す方法で管理可能な監査ファイルを作成します。

1 デフォルトの監査ポリシーを使用します。
具体的には、監査証跡へのイベントと監査トークンの追加を回避します。次のポリシーは、監査証跡のサイズに影響します。

- `arge` ポリシー – 環境変数を exec 監査イベントに追加します。
- `argv` ポリシー – コマンドパラメータを exec 監査イベントに追加します。
- `public` ポリシー – ファイルイベントを監査対象とする場合、公開ファイルで監査可能なイベントが発生するたびに、監査証跡にイベントを追加します。ファイルクラスには、`fa`、`fc`、`fd`、`fm`、`fr`、`fw`、`cl` などがあります。公開ファイルの定義については、600 ページの「監査の用語と概念」を参照してください。
- `path` ポリシー – `path` トークンを、省略可能な `path` トークンを含む監査イベントに追加します。
- `group` ポリシー – group トークンを、省略可能な `newgroups` トークンを含む監査イベントに追加します。
- `seq` ポリシー – sequence トークンをすべての監査イベントに追加します。
- `trail` ポリシー – trailer トークンをすべての監査イベントに追加します。
- `windata_down` ポリシー – Trusted Extensions で構成されたシステムで、ラベル付きウィンドウの情報がダウングレードされるときにイベントを追加します。

- windata_up ポリシー – Trusted Extensions で構成されたシステムで、ラベル付きウィンドウの情報がアップグレードされるときにイベントを追加します。
- zonename ポリシー – ゾーン名をすべての監査イベントに追加します。大域ゾーンが唯一の構成ゾーンである場合、zone, global をすべての監査イベントに追加します。

次の監査レコードは、ls コマンドの使用を示しています。ex クラスが監査対象で、デフォルトのポリシーが使用されています。

```
header,375,2,execve(2),,mach1,2009-08-06 11:19:57.388 -07:00
path,/usr/bin/ls
subject,jdoe,root,root,root,root,1401,737,0 0 mach1
return,success,0
```

すべてのポリシーがオンの場合、同じレコードが次のようになります。

```
header,375,2,execve(2),,mach1,2009-08-06 11:19:57.388 -07:00
path,/usr/bin/ls
attribute,100555,root,bin,136,432,0
exec_args,1,ls
exec_env,9,HOME=/,HZ=,LANG=C,LOGNAME=root,MAIL=/var/mail/root,PATH=/u
sr/sbin:/usr/bin,SHELL=/sbin/sh,TERM=xterm,TZ=US/Pacific
path,/lib/ld.so.1
attribute,100755,root,bin,136,4289,0
subject,jdoe,root,root,root,root,1401,737,0 0 mach1
group,root,other,bin,sys,adm,uucp,mail,tty,lp,nuucp,daemon
return,success,0
zone,global
sequence,313540
trailer,375
```

2 **audit_syslog.so プラグインを使用して、一部の監査イベントを syslog に送信します。**

この方法は、syslog ログに送信する監査イベントのバイナリレコードを保持する必要がない場合にのみ有効です。auditreduce コマンドを使用すると、レコードからバイナリファイルを取り除くことができるため、バイナリファイルのサイズが削減されます。

3 **特定のユーザーおよび役割について、監査イベントに audit_user ファイルを使用します。**

audit_control ファイルの監査クラスの数を減らすことにより、すべてのユーザーの監査の量を削減します。audit_user ファイルで、特定のユーザーおよび役割の監査クラスを追加します。

4 **独自のカスタマイズ監査クラスを作成します。**

使用しているシステムで監査クラスを作成できます。このクラスに、監視が必要な監査イベントをすべて指定します。手順については、632 ページの「監査クラスの追加方法」を参照してください。

注 – 既存の監査クラスの割り当てを変更する場合、新しいバージョンの Oracle Solaris OS にアップグレードするときに変更内容が失われることがありますインストールログを慎重に確認してください。

▼ ユーザーによるすべてのコマンドを監査する方法

サイトのセキュリティーポリシーの一環として、root ユーザーまたは管理役割により実行されるすべてのコマンドについて監査レコードが必要になることがあります。また、サイトによっては、ユーザーが実行するすべてのコマンドの監査レコードを必要とする場合もあります。

1 **lo クラスおよび ex クラスを監査します。**

ex クラスは、exec() 関数および execve() 関数のすべての呼び出しを監査します。lo クラスは、ログイン、ログアウト、および画面ロックを監査します。次の出力は、ex クラスおよび lo クラスのすべてのイベントを一覧表示します。

```
7:AUE_EXEC:exec(2):ps,ex
23:AUE_EXECVE:execve(2):ps,ex
...
6152:AUE_login:login - local:lo
6153:AUE_logout:logout:lo
6154:AUE_telnet:login - telnet:lo
6155:AUE_rlogin:login - rlogin:lo
6158:AUE_rshd:rsh access:lo
6159:AUE_su:su:lo
6162:AUE_rexecd:rexecd:lo
6163:AUE_passwd:passwd:lo
6164:AUE_rexd:rexd:lo
6165:AUE_ftpd:ftp access:lo
6171:AUE_ftpd_logout:ftp logout:lo
6172:AUE_ssh:login - ssh:lo
6173:AUE_role_login:role login:lo
6212:AUE_newgrp_login:newgrp login:lo
6213:AUE_admin_authenticate:admin login:lo
6221:AUE_screenlock:screenlock - lock:lo
6222:AUE_screenunlock:screenlock - unlock:lo
6227:AUE_zlogin:login - zlogin:lo
```

- 管理者についてこれらのクラスを監査するには、**audit_user** ファイルを変更します。

  次の例では、サイトが sysadm、 auditadm、 および netadm という 3 つの役割を作成しています。これらの役割と root アカウントは、exec クラスおよび lo クラスについて監査されます。

  ```
  ## audit_user file
  root:lo,ex:no
  sysadm:lo,ex:no
  auditadm:lo,ex:no
  netadm:lo,ex:no
  ```

- ユーザーに起因しないイベントについて **lo** クラスを監査するには、**audit_control** ファイルを変更します。

```
## audit_control file
...
naflags:lo
...
```

- すべてのユーザーについてこれらのクラスを監視するには、**audit_control** ファイルを変更します。

```
## audit_control file
flags:lo,ex
naflags:lo
...
```

出力は次のようになります。

```
header,375,2,execve(2),,mach1,2009-08-06 11:19:57.388 -07:00
path,/usr/bin/ls
subject,jdoe,root,root,root,root,1401,737,0 0 mach1
return,success,0
```

2 コマンドの引数を記録するには、**argv** ポリシーを設定します。

```
## audit_startup script
...
auditconfig -setpolicy +argv
...
```

exec_args トークンは、コマンド引数を記録します。

```
header,375,2,execve(2),,mach1,2009-08-06 11:19:57.388 -07:00
path,/usr/bin/ls
exec_args,1,ls
subject,jdoe,root,root,root,root,1401,737,0 0 mach1
return,success,0
```

3 コマンドの実行環境を記録するには、**arge** ポリシーを設定します。

```
## audit_startup script
...
auditconfig -setpolicy +arge
...
```

exec_env トークンは、コマンド環境を記録します。

```
header,375,2,execve(2),,mach1,2009-08-06 11:19:57.388 -07:00
path,/usr/bin/ls
exec_env,9,HOME=/,HZ=,LANG=C,LOGNAME=root,MAIL=/var/mail/root,
  PATH=/usr/sbin:/usr/bin,SHELL=/sbin/sh,TERM=xterm,TZ=US/Pacific
subject,jdoe,root,root,root,root,1401,737,0 0 mach1
return,success,0
```

4 引数とコマンド環境を記録するには、両方のポリシーを設定します。

```
## audit_startup script
...
```

```
auditconfig -setpolicy +argv
auditconfig -setpolicy +arge
...
```

出力は次のようになります。

```
header,375,2,execve(2),,mach1,2009-08-06 11:19:57.388 -07:00
path,/usr/bin/ls
exec_args,1,ls
exec_env,9,HOME=/,HZ=,LANG=C,LOGNAME=root,MAIL=/var/mail/root,
  PATH=/usr/sbin:/usr/bin,SHELL=/sbin/sh,TERM=xterm,TZ=US/Pacific
subject,jdoe,root,root,root,root,1401,737,0 0 mach1
return,success,0
```

▼ 特定のファイルに対する変更の監査レコードを検索する方法

/etc/passwd や /etc/default ディレクトリ内のファイルなど、限られた数のファイルに対するファイル書き込みを記録する場合、auditreduce コマンドを使用してファイルを見つけます。

1 **fw クラスを監査します。**

audit_user ファイルにクラスを追加すると、audit_control ファイルにクラスを追加する場合よりも、生成されるレコードが少なくなります。

- **fw クラスを audit_user ファイルに追加します。**

  ```
  ## audit_user file
  root:fw:no
  sysadm:fw:no
  auditadm:fw:no
  netadm:fw:no
  ```

- **fw クラスを audit_control ファイルに追加します。**

  ```
  ## audit_control file
  flags:lo,fw
  ...
  ```

2 特定のファイルの監査レコードを検索するには、**auditreduce** コマンドを使用します。

```
# /usr/sbin/auditreduce -o file=/etc/passwd,/etc/default -O filechg
```

auditreduce コマンドは、file 引数のすべてのインスタンスについて監査証跡を検索します。このコマンドにより、接尾辞 filechg を持つバイナリファイルが作成されます。このファイルには、必要なファイルのパス名を含むすべてのレコードが含まれています。-o file=*pathname* オプションの構文については、auditreduce(1M) のマニュアルページを参照してください。

3 **filechg** ファイルを読み取るには、**praudit** コマンドを使用します。

```
# /usr/sbin/praudit *filechg
```

▼ ユーザーの事前選択マスクを変更する方法

audit_control ファイルまたは ファイルを変更する場合、すでにログインしているユーザーの事前選択マスクは変更されません。事前選択マスクを強制的に変更する必要があります。

始める前に 監査を有効にしてユーザーがログインしてから、audit_control ファイルの flags または naflags の値を変更しました。新しく選択された監査クラスの監査対象とするために、すでにログインしているユーザーが必要です。

1 すでにログインしているユーザーの事前選択マスクを更新します。

2 つの選択肢があります。既存のセッションを終了するか、auditconfig コマンドを使用してユーザーの事前選択マスクを更新します。

- ユーザーの既存のセッションを終了します。

  ユーザーがログアウトしてふたたびログインするか、管理者がアクティブなセッションを手動で終了できます。新しいセッションでは、新しい事前選択マスクが継承されます。ただし、ユーザーの終了が実用的でない場合もあります。

- 各ユーザーの事前選択マスクを動的に変更します。

  audit_control ファイルの flags 属性が lo から lo,ex に変更されたものとします。

  a. ユーザーの監査 ID および監査セッション ID を決定します。

  まず、すべての通常ユーザーを検索します。次の例では、管理者が、root、daemon、 または lp により所有されていないすべてのプロセスを検索します。

```
# /usr/bin/pgrep -v -u root,daemon,lp | more
..
3941
3948
3949
10640 ...
```

  次に、ユーザーのプロセスの 1 つを使用して、ユーザーの監査 ID を検索します。

```
# auditconfig -getpinfo 3941
audit id = jdoe(1002)
process preselection mask = lo(0x1000,0x1000)
terminal id (maj,min,host) = 9426,65559,mach1(192.168.123.234)
audit session id = 713
```

  ユーザーの事前選択マスクには、lo クラスが含まれますが、新しく追加された ex クラスは含まれません。

ユーザーの監査 ID は 1002 です。ユーザーの監査セッション ID は 713 です。

2 ユーザーの事前選択マスクを変更します。
次の 2 つの方法のいずれかを使用します。

- ユーザーの監査セッション ID を使用して、ユーザーの事前選択マスクを変更します。

```
# /usr/sbin/auditconfig -setsmask lo,ex 713
```

- ユーザーの監査 ID を使用して、ユーザーの事前選択マスクを変更します。

```
# /usr/sbin/auditconfig -setumask lo,ex 1002
```

3 事前選択マスクが変更されたことを確認します。

```
# auditconfig -getpinfo 3941
audit id = jdoe(1002)
process preselection mask = ex,lo(0x40001000,0x40001000)
terminal id (maj,min,host) = 9426,65559,mach1(192.168.123.234)
audit session id = 713
```

▼ 特定のイベントが監査されないようにする方法

メンテナンスのために、監査イベントが監査されないようにする必要が生じることがあります。

1 イベントのクラスを no クラスに変更します。
たとえば、イベント 26 および 27 は pm クラスに属しています。

```
## audit_event file
...
25:AUE_VFORK:vfork(2):ps
26:AUE_SETGROUPS:setgroups(2):pm
27:AUE_SETPGRP:setpgrp(2):pm
28:AUE_SWAPON:swapon(2):no
...
```

これらのイベントを no クラスに変更します。

```
## audit_event file
...
25:AUE_VFORK:vfork(2):ps
26:AUE_SETGROUPS:setgroups(2):no
27:AUE_SETPGRP:setpgrp(2):no
28:AUE_SWAPON:swapon(2):no
...
```

pm クラスが現在監査中である場合でも、既存のセッションはイベント 26 および 27 を監査します。これらのイベントの監査を停止するには、ユーザーの事前選択マスクを更新する必要があります。

注意 – audit_event ファイルではイベントをコメントにしないでください。このファイルは、praudit コマンドがバイナリ監査ファイルを読み取るときに使用します。また、このファイルに一覧表示されたイベントが、保管された監査ファイルに含まれることがあります。

2 ユーザーの事前選択マスクを更新するには、669 ページの「ユーザーの事前選択マスクを変更する方法」の手順に従います。

▼ バイナリ監査ファイルのサイズを制限する方法

バイナリ監査ファイルは無制限に増大します。保管や検索を容易にするために、サイズの制限が必要となることがあります。元のファイルから小さいバイナリファイルを作成することもできます。

1 **Solaris 10 10/08** リリース以降、個々のバイナリ監査ファイルのサイズを制限するには **p_fsize** 属性を使用します。

audit_binfile.so プラグインの p_fsize 属性により、監査ファイルのサイズを制限できます。デフォルト値はゼロ (0) で、この場合はファイルが無制限に増大します。値は、512,000 から 2,147,483,647 までのバイト数で指定します。指定したサイズに達すると、現在の監査ファイルが閉じられ、新しいファイルが開きます。

次の例では、監査ファイルのサイズを 1M バイトに制限します。

```
plugin:name=audit_binfile.so; p_dir:/var/audit; p_fsize=1024000
```

2 **auditreduce** コマンドを使用して、レコードを選択し、これらのレコードを詳細分析用にファイルに書き込みます。

auditreduce -*lowercase* オプションは特定のレコードを検索します。

auditreduce -*Uppercase* オプションは選択したレコードをファイルに書き込みます。詳細は、auditreduce(1M) のマニュアルページを参照してください。

▼ ほかの OS からのログインを監査する方法

Oracle Solaris はソースに関係なく、すべてのログインを監査できます。

- ユーザーに起因するイベントおよび起因しないイベントの **lo** クラスを監査します。

  このクラスは、ログイン、ログアウト、および画面ロックを監査します。

```
## audit_control file
flags:lo
naflags:lo
...
```

注 – ssh ログインを監査するには、使用している Oracle Solaris システムで Oracle Solaris ssh デーモンを実行する必要があります。このデーモンは、Oracle Solaris 監査用に変更されます。詳細は、364 ページの「Oracle Solaris Secure Shell と OpenSSH プロジェクト」を参照してください。

▼ FTP および SFTP ファイル転送を監査する方法

FTP サービスは、ファイル転送のログを作成します。SSH プロトコルで実行する SFTP サービスは、Oracle Solaris 監査で監査できます。Oracle Solaris 監査では、両方のサービスへのログインを監査できます。

1 **FTP サービスのファイル転送とコマンドのログを作成するには、`ftpaccess(4)` のマニュアルページを参照してください。**

使用可能なログ作成オプションについては、「ログ作成機能」に関する節を参照してください。特に、有用なログを作成できるのは、`log commands` オプションおよび `log transfers` オプションです。

2 **`sftp` ファイル転送のログを作成するには、次の方法のいずれかまたは両方を実行します。**

- ファイル読み取りを監査します。

  SSH 接続によるファイル転送は、`sftp` コマンドを使用します。これらの転送は、`+fr` 監査フラグを使用して記録できます。失敗した `sftp` ファイル転送を監査するには、`-fr` 監査フラグを監査します。

  成功した `sftp` セッションの出力は次のとおりです。

```
header,138,2,open(2) - read,,ma2,2009-08-25 14:48:58.770 -07:00
path,/home/jdoe/vpn_connect
attribute,100644,jdoe,staff,391,437,0
subject,jdoe,jdoe,staff,jdoe,staff,4444,120289379,8457 65558 ma1
return,success,6
```

- 冗長オプションを **`sftp`** コマンドに使用します。

  `-v` オプションは 3 回まで繰り返すことができます。

```
# sftp -vvv [ other options ] hostname
```

3 **FTP および SFTP サービスへのアクセスを記録するには、`lo` クラスを監査します。**

次の出力に示すとおり、`ftpd` デーモンへのログインおよびログアウトで監査レコードが生成されます。

```
% bsmrecord -c lo | more
...
in.ftpd
  program     /usr/sbin/in.ftpd    See ftp access
```

```
  event ID    6165                AUE_ftpd
  class       lo                  (0x00001000)
      header
      subject
      [text]                      error message
      return

in.ftpd
  program     /usr/sbin/in.ftpd   See ftp logout
  event ID    6171                AUE_ftpd_logout
  class       lo                  (0x00001000)
      header
      subject
      return
...
```

SSH ログインは、`sftp` コマンドへのすべてのアクセスを記録します。

```
...
/usr/lib/ssh/sshd
  program     /usr/lib/ssh/sshd   See login - ssh
  event ID    6172                AUE_ssh
  class       lo                  (0x00001000)
      header
      subject
      [text]                      error message
      return
```

◆ ◆ ◆ 第 31 章

Oracle Solaris 監査 (参照)

この章では、Oracle Solaris 監査の重要なコンポーネントを説明します。この章の内容は次のとおりです。

Oracle Solaris 監査の概要は、第 28 章「Oracle Solaris 監査 (概要)」を参照してください。計画の提案については、第 29 章「Oracle Solaris 監査の計画」を参照してください。ご使用のシステムで監査を設定する手順については、第 30 章「Oracle Solaris 監査の管理 (手順)」を参照してください。

監査コマンド

この節では、次の各コマンドについての情報を提供します。

auditd デーモン

次のリストは、auditd デーモンのタスクの概要を示します。

- audit_control ファイルに指定されているディレクトリ内の監査ファイルを開いたり閉じたりします。ファイルは、記述した順序で開かれます。
- 1 つ以上のプラグインを読み込みます。Sun では 2 つのプラグインを提供します。audit_binfile.so プラグインは、バイナリ監査データをファイルに書き込みます。audit_syslog.so プラグインは、選択した監査レコードの概要テキストを syslog ログに渡します。
- カーネルから監査データを読み取り、auditd プラグインを使用してデータを出力します。
- audit_warn スクリプトを実行して、さまざまな状況を警告します。audit_binfile.so プラグインは、audit_warn スクリプトを実行します。デフォルトでは、このスクリプトは audit_warn 電子メールエイリアスとコンソールに警告を送信します。syslog.so プラグインは、audit_warn スクリプトを実行しません。
- デフォルトでは、監査ディレクトリがすべていっぱいになると、監査レコードを生成するプロセスは中断されます。また、auditd デーモンは、コンソールと audit_warn 電子メールエイリアスにメッセージを送ります。この時点では、システム管理者だけが、監査サービスの修復を行えます。管理者は、ログインして監査ファイルをオフラインメディアに書き込んだり、監査ファイルをシステムから削除したり、その他のクリーンアップ作業を実行したりできます。

  この監査ポリシーは、auditconfig コマンドを使用して構成し直すことができます。

auditd デーモンは、システムがマルチユーザーモードでブートする際に自動的に起動されますが、コマンド行から起動することもできます。auditd デーモンが起動すると、デーモンは監査ファイルに必要な空き容量を計算します。

auditd デーモンは、作成する監査ファイルの場所として audit_control ファイル内の監査ディレクトリの一覧を使用します。デーモンは、このディレクトリの一覧へのポインタを、最初のディレクトリに位置付けます。auditd デーモンは、監査ファイルを作成する必要があるたびに、一覧内の最初の使用可能ディレクトリ内に監査ファイルを格納します。一覧は、auditd デーモンの現在のポインタ位置から始まります。このポインタを一覧の最初のディレクトリに設定し直すには、audit -s コマンドを実行します。audit -n コマンドは、新しい監査ファイルに切り替えるように監査デーモンに指示します。新しいファイルは、現在のファイルと同じディレクトリ内に作成されます。

audit コマンド

audit コマンドは、auditd デーモンの動作を制御します。audit コマンドは、次の操作を実行できます。

- 監査機能を使用可能および使用不可にします
- auditd デーモンを設定し直します
- ローカルシステム上のプロセス事前選択マスクを調整します
- 監査レコードを別の監査ファイルに書き込みます

使用可能なオプションについては、audit(1M) のマニュアルページを参照してください。

bsmrecord コマンド

bsmrecord コマンドは、/etc/security/audit_event ファイル内に定義されている監査イベントの書式を表示します。監査イベントの監査 ID、監査クラス、監査フラグ、およびレコードの監査トークンが順に出力されます。オプションを指定しなかった場合、bsmrecord 出力は端末に表示します。-h オプションを指定した場合、ブラウザでの表示に適した形式で出力します。bsmrecord コマンドの使用例については、651 ページの「監査レコードの書式の表示方法」を参照してください。また、bsmrecord(1M) のマニュアルページも参照してください。

auditreduce コマンド

auditreduce コマンドは、バイナリ形式で格納されている監査レコードをまとめます。コマンドを実行すると、1 つまたは複数の入力監査ファイルから監査レコードがマージできます。このコマンドでは、監査レコードの事後選択を実行することもできます。レコードはバイナリ形式のままです。監査トレール全体をマージするには、監査サーバー上でこのコマンドを実行します。監査サーバーとは、すべての監査ファイルシステムがマウントされているシステムのことです。詳細は、auditreduce(1M) のマニュアルページを参照してください。

auditreduce コマンドを使用すると、複数のシステム上のすべての監査対象動作を 1 か所から追跡できます。このコマンドは、すべての監査ファイルを論理的に結合し、単一の監査トレールとして読み取ることができます。サイト内のすべてのシステムが同一の監査構成を持つようにするとともに、サーバーと監査ファイル用のローカルディレクトリを作成しておく必要があります。auditreduce では、レコードの生成方法や格納場所は無視されます。オプションを指定しなかった場合、auditreduce コマンドは、監査ルートディレクトリのすべてのサブディレクトリ内のすべての監査ファイルの監査レコードをマージします。通常、/etc/security/audit が監査ルートディレクトリです。auditreduce コマンドは、マージ結果を標準出力に送ります。マージ結果は、時系列に並べて 1 つの出力ファイルに格納することもできます。このファイルの形式はバイナリデータです。

auditreduce コマンドを使用して、特定の種類のレコードを選択し、解析に利用することもできます。auditreduce コマンドのマージ機能と選択機能は、論理的に互いに依存しません。auditreduce コマンドは、入力ファイルのレコードを読み取ると、マージしてディスクに書き込む前に、データを抽出します。

auditreduce コマンドにオプションを指定すると、次の操作も実行できます。

- 指定された監査クラスによって生成された監査レコードを要求します
- 特定のユーザーによって作成された監査レコードを要求します
- 特定の日付に作成された監査レコードを要求します

auditreduce に引数を指定しなかった場合は、デフォルトの監査ルートディレクトリ /etc/security/audit 内のサブディレクトリが検査されます。このコマンドは、*start-time.end-time.hostname* ファイルが配置されている files ディレクトリを検査します。auditreduce コマンドは、監査データが異なるディレクトリに格納されている場合に非常に有用です。図 31–1 は、監査データがホスト別のディレクトリ内に格納されている場合を示しています。図 31–2 は、監査データが監査サーバー別のディレクトリ内に格納されている場合を示しています。

図 31–1 ホストごとに格納された監査トレール

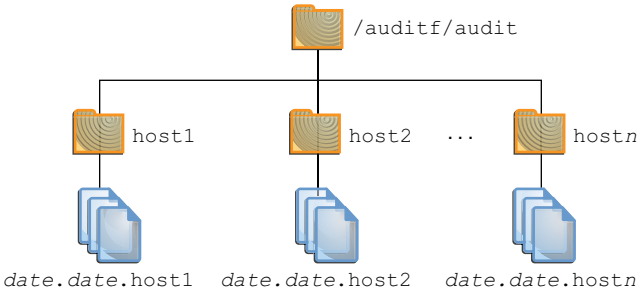
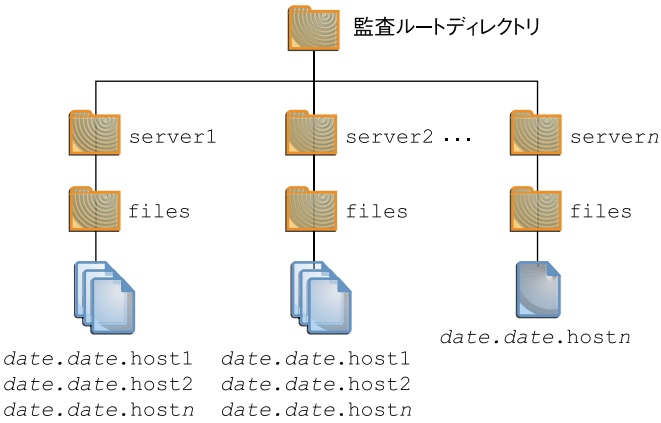


図 31–2 サーバーごとに格納された監査トレール



/etc/security/audit のパーティションが小さい場合、デフォルトのディレクトリに監査データを格納しない方法もあります。-R オプションを使用して、auditreduce コマンドを別のディレクトリに渡すことができます。

```
# auditreduce -R /var/audit-alt
```

-S オプションを使用して、特定のサブディレクトリを指定することもできます。

```
# auditreduce -S /var/audit-alt/host1
```

その他のオプションや例については、auditreduce(1M) のマニュアルページを参照してください。

praudit コマンド

praudit コマンドは、auditreduce コマンドのバイナリ出力をユーザーが読めるようにします。praudit コマンドは、標準入力からバイナリ形式の監査レコードを読み込み、そのレコードを表示可能な書式で表示します。auditreduce コマンドまたは 1 つの監査ファイルからの出力は、praudit コマンドの入力にパイプできます。cat コマンドを使用すると、複数のファイルを連結して入力にパイプすることができます。tail コマンドを使用すると、現在の監査ファイルを入力にパイプできます。

praudit コマンドでは、次の 4 つの出力形式を生成できます。5 番目のオプション -l (長形式) では、出力の 1 行に 1 つの監査レコードが表示されます。デフォルトでは、出力の 1 行につき 1 つの監査トークンが表示されます。-d オプションを指定すると、トークンフィールドおよびトークン間で使用される区切り文字を変更できます。デフォルトの区切り文字は、コンマです。

- デフォルト – オプションを指定しないで **praudit** コマンドを実行すると、1 行につき 1 つの監査トークンが表示されます。監査イベントは `ioctl(2)` システムコールなどのその内容が表示されます。テキストで表示できる値はすべてテキスト形式で表示されます。たとえば、ユーザーは、ユーザー ID ではなく、ユーザー名で表示されます。
- –r オプション – このオプションでは、数値で表現できる値はすべて数値として表示されます。たとえば、ユーザーはユーザー ID で、インターネットアドレスは 16 進形式で、モードは 8 進形式で表示されます。監査イベントは、イベント番号 (158 など) で表示されます。
- –s オプション – このオプションでは、監査イベントがテーブル名 (**AUE_IOCTL**) で表示されます。その他のトークンは、デフォルトと同じ形式で表示されます。
- –x オプション – このオプションでは、監査レコードが XML 形式で表示されます。このオプションは、出力結果をブラウザや XML を操作するスクリプトに入力する場合に便利です。

  XML は、監査サービスが提供する DTD によって記述されます。また、Oracle Solaris ソフトウェアにはスタイルシートも付属しています。DTD とスタイルシートは `/usr/share/lib/xml` ディレクトリ内に格納されています。

`praudit` コマンドのデフォルトの出力形式では、各レコードは監査トークンの並びとして容易に識別できます。各トークンは 1 行ごとに出力されます。各監査レコードは `header` トークンで始まります。`awk` コマンドなどを使用すると、出力をさらに処理できます。

次の出力は、`header` トークンを `praudit -l` コマンドで出力したものです。

```
header,173,2,settppriv(2),,example1,2010-10-10 10:10:02.020 -07:00
```

次の出力は、同じ `header` トークンを `praudit -r` コマンドで出力したものです。

```
121,173,2,289,0x0000,192.168.86.166,1066077962,174352445
```

例 31–1　praudit 出力をスクリプトで処理する

`praudit` コマンドの出力は、必要に応じてテキストとして操作できます。たとえば、`auditreduce` コマンドでは選択できないレコードを選択することがあります。単純なシェルスクリプトを使用すると、`praudit` コマンドの出力を処理できます。次の単純なスクリプトの例は、1 つの監査レコードを 1 行にまとめ、ユーザーが指定した文字列を検索し、最後に監査ファイルを元の形式に戻します。

```
#!/bin/sh
#
## This script takes an argument of a user-specified string.
#  The sed command prefixes the header tokens with Control-A
#  The first tr command puts the audit tokens for one record
#  onto one line while preserving the line breaks as Control-A
#
praudit | sed -e '1,2d' -e '$s/^file.*$//' -e 's/^header/^aheader/' \\
```

例 31–1 praudit 出力をスクリプトで処理する (続き)

```
| tr '\\012\\001' '\\002\\012' \\
| grep "$1" \\       Finds the user-specified string
| tr '\\002' '\\012'       Restores the original newline breaks
```

スクリプトの ^a は、^ と a という 2 つの文字ではなく、Control-A です。この接頭辞によって、header トークンが、テキストとして表示される header という文字列と区別されます。

auditconfig コマンド

auditconfig コマンドは、監査構成パラメータを取得および設定するためのコマンド行インタフェースを提供します。auditconfig コマンドは、次の操作を実行できます。

- 監査ポリシーの表示、チェック、構成
- 監査状態 (オン/オフ) の確認
- 監査ディレクトリと監査ファイルの管理
- 監査キューの管理
- 事前選択マスクの取得/設定
- 監査イベントの監査クラスへのマッピングの取得/設定
- セッション ID や監査 ID などの構成情報の取得/設定
- プロセス、シェル、セッションの監査特性の構成
- 監査統計情報のリセット

コマンドオプションの詳細は、auditconfig(1M) のマニュアルページを参照してください。

監査サービスで使用されるファイル

監査サービスでは、次のファイルが使用されます。

- 682 ページの「system ファイル」
- 682 ページの「syslog.conf ファイル」
- 682 ページの「audit_class ファイル」
- 682 ページの「audit_control ファイル」
- 684 ページの「audit_event ファイル」
- 684 ページの「audit_startup スクリプト」
- 685 ページの「audit_user データベース」
- 686 ページの「audit_warn スクリプト」
- 687 ページの「bsmconv スクリプト」

system ファイル

/etc/system ファイルには、カーネルが初期設定で読み込み、システム動作をカスタマイズするためのコマンドが格納されます。bsmconv および bsmunconv シェルスクリプトは、監査機能を起動および終了するときに使用され、/etc/system ファイルを変更します。bsmconv シェルスクリプトは、/etc/system ファイルに次の行を追加します。

```
set c2audit:audit_load=1
```

set c2audit:audit_load=1 エントリは、システムのブート時に監査用のカーネルモジュールをロードします。bsmunconv シェルスクリプトは、システムのリブート時に監査を無効にします。このコマンドは、/etc/system ファイルから c2audit の行を削除します。

syslog.conf ファイル

/etc/syslog.conf ファイルは audit_syslog.so プラグインと一緒に機能して、監査レコードをテキスト形式で格納します。syslog ユーティリティーが監査レコードを格納できるように、syslog.conf ファイルを設定できます。例は、627 ページの「syslog 監査ログの構成方法」を参照してください。

audit_class ファイル

/etc/security/audit_class ファイルは、監査クラスを定義します。監査クラスは、監査イベントのグループです。audit_control ファイル内のこのクラス名を使用して、監査するイベントのクラスを事前選択します。クラスには、失敗したイベントだけ、または正常なイベントだけを選択する接頭辞を使用できます。詳細は、690 ページの「監査クラスの構文」を参照してください。

スーパーユーザーまたはそれと同等の役割を持つ管理者は、監査クラスの定義を変更できます。管理者は、audit_class ファイルをテキストエディタで編集することによって、新しい監査クラスを定義したり、既存クラスの名前を変更したり、既存クラスにその他のさまざまな変更を施したりすることができます。詳細は、audit_class(4) のマニュアルページを参照してください。

audit_control ファイル

各システム上の/etc/security/audit_control ファイルには、auditd デーモンの構成情報が含まれます。このファイルを使用すると、すべてのシステムが、その監査レコードを格納する遠隔監査ファイルシステムをマウントできるようになります。

audit_control ファイルには、次の 5 種類の情報を指定できます。各行の情報は、キーワードで始まります。

- flags キーワード – このキーワードで始まるエントリは、システム上のすべてのユーザーを対象に監査するイベントのクラスを事前選択します。ここで指定する監査クラスは、「システム全体の監査事前選択マスク」を指定します。監査クラスはコンマで区切ります。
- naflags キーワード – このキーワードで始まるエントリは、特定のユーザーに起因しない動作が発生したときに監査するイベントのクラスを事前選択します。監査クラスはコンマで区切ります。na イベントクラスは、このエントリにあります。naflags エントリを使用すると、通常はユーザーに起因するがユーザーに起因できないほかのイベントクラスのログをとることができます。たとえば、ブート時に起動するプログラムがファイルを読み取ると、naflags エントリ内の fr がそのイベントのレコードを作成します。
- minfree キーワード – このキーワードは非推奨です。audit_binfile.so プラグインの p_minfree 属性を使用します。

  p_minfree 属性は、すべての監査ファイルシステムについて、最小空き容量レベルをパーセントで定義します。この割合は、0 以上で指定する必要があります。デフォルトは 20% です。監査ファイルシステムの使用率が 80% に達すると、次に利用可能な監査ディレクトリに監査データが格納されるようになります。詳細は、audit_warn(1M) のマニュアルページを参照してください。
- dir キーワード – このキーワードは非推奨です。audit_binfile.so プラグインの p_dir 属性を使用します。

  p_dir 属性は、ディレクトリの場所を一覧表示します。各行の値には、監査ファイルを格納するためにシステムが使用する、監査ファイルシステムとディレクトリを定義します。1 つまたは複数のディレクトリの場所を指定できます。値については、順番が重要になります。auditd デーモンは、ここで指定した順番でディレクトリに監査ファイルを作成します。1 番目のディレクトリがそのシステムの「1 次監査ディレクトリ」になり、2 番目のディレクトリが「2 次監査ディレクトリ」になります。1 番目のディレクトリがいっぱいになると、auditd デーモンは 2 番目以降のディレクトリに監査トレールファイルを作成します。詳細は、audit(1M) のマニュアルページを参照してください。
- plugin キーワード – プラグインモジュール audit_binfile.so および audit_syslog.so のプラグインパスを指定します。audit_binfile.so モジュールは、バイナリ監査ファイルの作成を処理します。audit_syslog.so モジュールは、Oracle Solaris 監査レコードをテキスト形式にリアルタイムで変換します。audit_syslog.so プラグインの p_flags 属性に指定された監査クラスは、事前選択された監査クラスのサブセットである必要があります。

audit_control ファイルの詳細は、audit_control(4) のマニュアルページを参照してください。プラグインについては、692 ページの「監査プラグイン」、audit_binfile(5) および audit_syslog(5) のマニュアルページを参照してください。

例 31–2　audit_control ファイルの例

次の例は、システム noddy で使用する audit_control ファイルです。noddy では、監査サーバー blinken 上で 2 つの監査ファイルシステムを使用し、2 つ目の監査

例 31–2 audit_control ファイルの例 (続き)

サーバー winken からマウントされる 3 つ目の監査ファイルシステムを使用します。3 つ目のファイルシステムは、blinken 上の監査ファイルシステムがいっぱいであるか使用できないときにだけ使用されます。minfree の値として 20% を指定しているため、ファイルシステムの使用率が 80% に達した時点で警告スクリプトが実行されます。次の設定では、監査対象としてログイン操作と管理操作が指定されています。これらの操作について、その成功と失敗が監査されます。ファイルシステムオブジェクト作成の失敗を除くすべての失敗が、監査対象となります。また、ユーザーに起因しないイベントも監査されています。syslog 監査ログはより少ない監査イベントを記録します。このログには、失敗したログインと失敗した管理操作のテキストサマリーが記録されます。

Solaris 10 リリースでは、dir 行および minfree 行は非推奨です。次の例では、plugin 行に改行が含まれていません。

```
flags:lo,am,-all,^-fc
naflags:lo,nt
plugin:name=audit_binfile.so; p_minfree=20; p_dir=/var/audit/blinken/files,
/var/audit/blinken.1/files,/var/audit/winken
plugin:name=audit_syslog.so; p_flags=-lo,-am
```

audit_event ファイル

/etc/security/audit_event ファイルには、監査イベントから監査クラスへのマッピングのデフォルト値が格納されます。このファイルを編集して、クラスのマッピングを変更できます。クラスのマッピングを変更したときは、システムをリブートするか、変更したマッピングをカーネルに読み込むために auditconfig -conf コマンドを実行する必要があります。詳細は、audit_event(4) のマニュアルページを参照してください。

audit_startup スクリプト

システムがマルチユーザーモードに移行すると、/etc/security/audit_startup スクリプトが監査サービスを自動的に構成します。auditd デーモンは、スクリプトが次のタスクを実行してから起動します。

- 監査イベントから監査クラスへのマッピングを構成する
- 監査ポリシーオプションを設定する

詳細は、audit_startup(1M) のマニュアルページを参照してください。

audit_user データベース

/etc/security/audit_user データベースは、システム全体の事前選択クラスを個々のユーザーごとに変更します。audit_user データベース内のユーザーエントリに追加するクラスは、audit_control ファイルにある設定を次の 2 つの方法で変更します。

- そのユーザーについて常に監査する監査クラスを指定する
- そのユーザーについて監査しない監査クラスを指定する

audit_user データベースの各ユーザーエントリには、次の 3 つのフィールドがあります。

username:*always-audit-classes*:*never-audit-classes*

監査フィールドは、順番に処理されます。

- *always-audit-classes* フィールドは、指定されたクラスの監査を有効にします。このフィールドを使用してシステム全体の設定を変更します。たとえば、*always-audit-classes* フィールドに all を指定すると、ユーザーのすべての動作が監査されます。
- *never-audit-classes* フィールドは、指定されたクラスの監査を無効にします。このフィールドを使用して、システム設定を上書きします。*never-audit-classes* フィールドに all を指定すると、audit_control ファイルに指定された監査クラスを含め、ユーザーのすべての監査がオフになります。

たとえば、ファイルシステムオブジェクトの正常な読み取り動作を除き、システム全体の監査設定を tamiko というユーザーに適用するとします。次の audit_user エントリでの 2 番目のコロン (:) の位置に注意してください。

```
tamiko:^+fr:no          modify system defaults for fr
```

前述のエントリは、「正常なファイル読み取り動作を除くすべての動作を監査する」ことを意味しています。

ユーザー tamiko について、正常なファイル読み取り動作を除くすべての動作を監査する場合、次のエントリを使用します。

```
tamiko:all,^+fr:no          audit everything except fr
```

ユーザー tamiko の正常なファイル読み取り動作について、システムのデフォルト設定を上書きするとします。次のエントリは、「常にすべての動作を監査するが、正常なファイルの読み取り動作はまったく監査しない」ことを意味しています。

```
tamiko:all:+fr          override system defaults for fr
```

注 – 正常終了したイベントと失敗したイベントは別々に取り扱われます。プロセスが生成する監査レコードの数は、イベントが正常終了した場合よりも失敗した場合のほうが多くなる可能性があります。

audit_warn スクリプト

auditd デーモンで監査レコードの書き込み中に異常な状態が発生すると、/etc/security/audit_warn スクリプトは電子メールエイリアスに通知します。このスクリプトをサイトに合わせてカスタマイズすることで、手動による対処が必要な状態を警告するようにしたり、そのような状態を自動的に処理するための方法を指定したりできます。エラーが発生すると、audit_warn スクリプトは、daemon.alert の重要度で syslog にメッセージを書き込みます。syslog.conf を使用すると、syslog メッセージのコンソール表示を設定できます。audit_warn スクリプトはさらに、audit_warn 電子メールエイリアスにもメッセージを送信します。このエイリアスは、監査構成の一部として設定します。

auditd デーモンは、次の条件を検出すると audit_warn スクリプトを起動し、audit_warn エイリアスに電子メールを送信します。

- 監査ディレクトリが minfree の許容値を超えていっぱいになった場合。minfree 値は弱い制限値で、監査ファイルシステム上で使用できる領域の割合です。

  audit_warn スクリプトは、文字列 soft と、使用可能領域が下限値を下回ったディレクトリ名を使用して起動されます。auditd デーモンは、次の適切なディレクトリに自動的に切り替えます。デーモンは、新しいディレクトリが minfree 制限値に達するまで、このディレクトリに監査ファイルを書き込みます。その後、auditd デーモンは、audit_control ファイルに指定された順序で残りの各ディレクトリに順次アクセスします。デーモンは、各ディレクトリが minfree 制限値に達するまで監査レコードを書き込みます。

- すべての監査ディレクトリが minfree しきい値に達した場合。

  文字列 allsoft を使用して audit_warn スクリプトが起動されます。コンソールにメッセージが出力されます。さらに、audit_warn のエイリアスに電子メールが送信されます。

  audit_control ファイルに指定されたすべての監査ディレクトリが minfree しきい値に達すると、auditd デーモンは最初のディレクトリに戻ります。デーモンは、そのディレクトリが完全にいっぱいになるまで監査レコードを書き込みます。

- 監査ディレクトリが完全にいっぱいになり、残りの容量がなくなった場合。

  文字列 hard とディレクトリ名を使用して、audit_warn スクリプトが起動されます。コンソールにメッセージが出力されます。さらに、audit_warn のエイリアスに電子メールが送信されます。

auditd デーモンは、使用可能領域が残っている次の適切なディレクトリに自動的に切り替えます。その後、auditd デーモンは、audit_control ファイルに指定された順序で残りの各ディレクトリに順次アクセスします。デーモンは、各ディレクトリがいっぱいになるまで完全レコードを書き込みます。

- すべての監査ディレクトリが完全にいっぱいになった場合。引数として文字列 allhard を使用して、audit_warn スクリプトが起動されます。

  デフォルトでは、コンソールにメッセージが書き込まれます。さらに、audit_warn のエイリアスに電子メールが送信されます。監査レコードを生成するはずのプロセスは引き続き発生しますが、監査レコードはカウントされます。監査レコードは生成されません。この状況に対処する方法の例については、例 30–16 と660 ページの「監査トレールのオーバーフローを防ぐ方法」を参照してください。
- 内部エラーが発生した場合。次のような内部エラーが考えられます。
  - ebusy – 別の auditd デーモンプロセスがすでに動作している
  - tmpfile – 一時ファイルを使用できない
  - postsigterm – 監査のシャットダウン中に信号を受信した
  - plugin *name* – プラグインの実行中にエラーが発生した
- audit_control ファイルの構文に問題が検出された場合。デフォルトでは、コンソールにメッセージが書き込まれます。さらに、audit_warn のエイリアスに電子メールが送信されます。

perzone 監査ポリシーが設定されている場合、非大域ゾーンの auditd のインスタンスがゾーンの audit_warn スクリプトを呼び出します。詳細は、audit_warn(1M) のマニュアルページを参照してください。

bsmconv スクリプト

/etc/security/bsmconv スクリプトは、監査サービスを有効にします。bsmunconv コマンドは、監査サービスを無効にします。bsmconv スクリプトを実行したあと、監査ディレクトリと監査構成ファイルを設定します。リブート時に、監査が有効になります。

詳細は、bsmconv(1M) のマニュアルページを参照してください。

監査を管理するための権利プロファイル

Oracle Solaris には、監査サービスを構成したり監査トレールを分析したりするための権利プロファイルが用意されています。

- **Audit Control** - 特定の役割が Oracle Solaris 監査をできるようにします。この権利プロファイルは、監査サービスが使用する構成ファイルを構成する権限を付与します。特定の役割が監査コマンドを実行するようにもします。Audit Control プロ

ファイルで割り当てられた役割が実行できるコマンドは、 `audit`、 `auditd`、 `auditconfig`、 `bsmconv`、 および `bsmunconv` です。

- **Audit Review** - 特定の役割が Oracle Solaris 監査レコードを分析できるようにします。この権利プロファイルは、`praudit` コマンドと `auditreduce` コマンドを使って監査レコードを読み取る権限を付与します。この権利プロファイルで割り当てられた役割は、`auditstat` コマンドを実行することもできます。
- **System Administrator** – Audit Review 権利プロファイルを含みます。System Administrator 権利プロファイルで割り当てられた役割は、監査レコードを分析できます。

監査サービスを扱う役割を設定する方法については、210 ページの「RBAC の構成 (作業マップ)」を参照してください。

監査と Oracle Solaris ゾーン

非大域ゾーンは、大域ゾーンの監査とまったく同様に監査することも、独自のフラグ、記憶領域、および監査ポリシーを設定することもできます。

すべてのゾーンが同様に監査される場合には、大域ゾーンの構成ファイルが、すべてのゾーンにおける監査の設定を提供します。`+zonename` ポリシーオプションが役に立ちます。このオプションが指定されていると、すべてのゾーンからの監査レコードには、ゾーンの名前が含まれます。監査レコードをゾーン名を使用して、事後選択することができます。監査ポリシーの詳細は、617 ページの「監査ポリシーの決定」を参照してください。例については、639 ページの「監査ポリシーを構成する方法」を参照してください。

ゾーンを個別に監査することもできます。ポリシーオプション `perzone` が大域ゾーンで指定されているとき、各非大域ゾーンは、それぞれの監査デーモンの実行、監査キューの処理、および監査レコードの内容と場所の指定を行います。非大域ゾーンは、ほとんどの監査ポリシーオプションを指定できます。システム全体に影響するポリシーを設定できないため、非大域ゾーンは `ahlt` または `perzone` ポリシーを指定できません。詳細については、607 ページの「Oracle Solaris ゾーンを含むシステムでの監査」および 612 ページの「ゾーン内の監査の計画方法」を参照してください。

ゾーンの詳細は、『Oracle Solaris のシステム管理 (Oracle Solaris コンテナ : 資源管理と Oracle Solaris ゾーン)』のパート II「ゾーン」を参照してください。

監査クラス

システム全体での Oracle Solaris 監査のデフォルト値は、1 つ以上のイベントのクラスを指定して事前選択されます。クラスは、システムの audit_control ファイルでシステムごとに事前選択されます。システムを使用するすべてのユーザーが、これらのイベントのクラスに対して監査されます。このファイルについては、682 ページの「audit_control ファイル」を参照してください。

監査クラスを構成し、新しい監査クラスを作成できます。監査クラス名は、最長 8 文字です。クラスの説明は、72 文字に制限されています。数値と英数字以外の文字が使用できます。

audit_user データベースのユーザーのエントリに監査クラスを追加して、ユーザーごとの監査対象を変更できます。監査クラスは、auditconfig コマンドの引数としても使用します。詳細は、auditconfig(1M) のマニュアルページを参照してください。

監査クラスの定義

次の表に、事前に定義されている監査クラス、監査クラスの記述名、および短い説明を示します。

表 31–1 事前に定義されている監査クラス

監査クラス	記述名	説明
all	all	すべてのクラス (メタクラス)
no	no_class	事前選択を無効にする空の値
na	non_attrib	ユーザーが原因ではないイベント
fr	file_read	データを読み取る、読み取りのために開く
fw	file_write	データを書き込む、書き込みのために開く
fa	file_attr_acc	オブジェクト属性にアクセスする: stat、 pathconf
fm	file_attr_mod	オブジェクト属性を変更する: chown、 flock
fc	file_creation	オブジェクトの作成
fd	file_deletion	オブジェクトの削除
cl	file_close	close システムコール
ap	application	アプリケーションが定義するイベント
ad	administrative	管理上の操作 (旧 administrative メタクラス)
am	administrative	管理上の操作 (メタクラス)

表 31–1 事前に定義されている監査クラス (続き)

監査クラス	記述名	説明
ss	system state	システムの状態を変更
as	system-wide administration	システム全体の管理
ua	user administration	ユーザー管理
aa	audit administration	監査の使用
ps	process start	プロセスの起動、プロセスの停止
pm	process modify	プロセスの変更
pc	process	プロセス (メタクラス)
ex	exec	プログラムの実行
io	ioctl	ioctl() システムコール
ip	ipc	System V IPC 操作
lo	login_logout	ログインとログアウトのイベント
nt	network	ネットワークイベント: bind、 connect、 accept
ot	other	その他、デバイス割り当てや memcntl() など

/etc/security/audit_class ファイルを変更して、新しいクラスを定義できます。既存クラスの名前の変更も可能です。詳細は、audit_class(4) のマニュアルページを参照してください。

監査クラスの構文

成功した場合に監査されるイベント、失敗した場合に監査されるイベント、または両方の場合に監査されるイベントがあります。接頭辞を指定しなかったイベントのクラスは、成功した場合も失敗した場合も監査されます。プラス (+) 接頭辞が付いた場合、イベントのクラスは成功した場合のみが監査されます。マイナス (-) 接頭辞が付く場合、イベントのクラスは失敗した場合のみが監査されます。次の表に、監査クラスの例をいくつか示します。

表 31–2 接頭辞 (プラス記号、マイナス記号) の付いた監査クラス

[*prefix*] *class*	意味
lo	成功したすべてのログインとログアウト、および失敗したすべてのログインを監査します。ログアウトが失敗することはありません。
+lo	成功したすべてのログインとログアウトを監査します。

表 31–2 接頭辞 (プラス記号、マイナス記号) の付いた監査クラス (続き)

[*prefix*] *class*	意味
`-all`	失敗したすべてのイベントを監査します。
`+all`	成功したすべてのイベントを監査します。

注意 – `all` クラスを指定すると、大量のデータが生成され、監査ファイルシステムがすぐにいっぱいになる可能性があります。`all` クラスは、特別な理由ですべての活動を監査する場合にだけ使用してください。

キャレット接頭辞 ^ を使用すると、選択されている監査フラグをさらに変更できます。次の表は、キャレット接頭辞を使って選択済みの監査クラスを変更する方法を示したものです。

表 31–3 指定済みの監査クラスを変更するキャレット接頭辞

^[*prefix*]*class*	意味
`-all,^-fc`	失敗したすべてのイベントを監査します。ただし、失敗したファイルシステムオブジェクト作成は監査しません
`am,^+aa`	成功または失敗したすべての管理イベントを監査します。ただし、成功した監査管理は監査しません
`am,^ua`	成功または失敗したすべての管理イベントを監査します。ただし、ユーザー管理イベントは監査しません

監査クラスとその接頭辞は、次のファイルとコマンドで使用できます。

- `audit_control` ファイルの `flags` 行
- `audit_control` ファイルの `plugin:name=audit_syslog.so; p_flags=` 行
- `audit_user` データベース内のユーザーのエントリ
- `auditconfig` コマンドオプションの引数として

`audit_control` ファイル内での接頭辞の使用例については、682 ページの「`audit_control` ファイル」を参照してください。

監査プラグイン

監査プラグインでは、監査キューの監査レコードの処理方法を指定します。監査プラグインである audit_binfile.so および audit_syslog.so は、audit_control ファイル内で指定されます。このプラグインとパラメータを使って、次の内容を指定できます。

- audit_binfile.so プラグイン、p_dir パラメータにより、バイナリデータの送信先
- audit_binfile.so プラグイン、p_minfree パラメータにより、管理者への警告の基準となるディスクの最小空き容量
- audit_binfile.so プラグイン、p_fsize パラメータにより、監査ファイルの最大サイズ

  p_fsize パラメータプラグインは、Solaris 10 10/08 リリース以降使用可能
- audit_syslog.so プラグイン、p_flags パラメータにより、syslog に送信する監査レコードの選択
- qsize パラメータにより、そのプラグインでキューに入る監査レコードの最大数

audit_binfile(5)、 audit_syslog(5)、 および audit_control(4) のマニュアルページを参照してください。

監査ポリシー

監査ポリシーには、監査トレールにトークンまたは情報を追加するかどうかを指定します。

arge、 argv、 group、 path、 seq、 trail、 windata_down、 windata_up、 および zonename ポリシーは、監査レコードにトークンを追加します。

その他のポリシーでは、トークンは追加されません。ahlt および cnt ポリシーはカーネル監査レコードが配信できない場合の動作を決定、public ポリシーは公開ファイルの監査を制限、perzone ポリシーは非大域ゾーンの別個の監査キューを確立します。

さまざまな監査ポリシーオプションの働きについては、617 ページの「監査ポリシーの決定」を参照してください。監査ポリシーオプションについては、auditconfig(1M) のマニュアルページの -setpolicy オプションを参照してください。使用可能なポリシーオプションのリストについては、auditconfig -lspolicy コマンドを実行します。

プロセスの監査特性

最初のログイン時に次の監査特性が設定されます。

- 「プロセス事前選択マスク」– **audit_control** ファイルと `audit_user` データベースの監査クラスを結合したもの。ユーザーがログインすると、ログインプロセスは、事前に選択されたクラスを結合し、そのユーザーのプロセスに対する「プロセス事前選択マスク」を確立します。プロセス事前選択マスクは、各監査クラス内のイベントで監査レコードを生成するかどうかを指定します。

  ユーザーのプロセス事前選択マスクを取得する方法は、次のアルゴリズムで表されます。

  (`flags` *line* + *always-audit-classes*) - *never-audit-classes*

  `audit_control` ファイル内の `flags` 行にある監査クラスを、*audit_user* データベース内のユーザーエントリの `always-audit-classes` フィールドにあるクラスに追加します。次に、ユーザーの *never-audit-classes* フィールドにあるクラスを全体のクラスから減算します。
- 「監査 ID」– ユーザーがログインすると、プロセスは監査 ID を取得します。監査 ID は、ユーザーの初期プロセスが起動するすべての子プロセスに継承されます。監査 ID はアカウントの追跡を強行するときにも役立ちます。ユーザーが `root` になったあとも、監査 ID はそのまま変わらずに残ります。各監査レコード内に保存された監査 ID を使用すると、常に動作を追跡してログインした元のユーザーまでたどることができます。
- 「監査セッション ID」– 監査セッション ID はログイン時に割り当てられます。このセッション ID はすべての子プロセスに継承されます。
- 「端末 ID (ポート ID、マシンアドレス)」 – 端末 ID は、ホスト名とインターネットアドレスで構成され、そのあとにユーザーがログインした物理デバイスを識別する一意の番号が続きます。通常、ログインはコンソールから行われ、そのコンソールデバイスに対応する番号は `0` になります。

監査トレール

「監査トレール」はバイナリ監査ファイルを含み、`auditd` デーモンによって作成されます。`bsmconv` コマンドにより監査サービスが有効になると、システムの起動時に `auditd` デーモンが起動します。`auditd` デーモンは、監査トレールデータを収集し、監査レコードを書き込みます。

監査レコードは、監査ファイル専用のシステム上にバイナリ形式で格納されます。監査ディレクトリは、監査専用でないほかのファイルシステム内に物理的に配置することもできますが、予備のディレクトリを除き、この配置は行わないでください。予備のディレクトリとは、他の適切なディレクトリが使用できないときに限り、監査ファイルが書き込まれるディレクトリです。

監査ディレクトリを監査専用でないファイルシステムに配置しても構わない場合が、もう 1 つあります。つまり、ソフトウェア開発環境を使用していて、監査が任意である場合は、そうしても構いません。監査トレールを保存するよりも、ディスク容量を有効に使用するほうが重視されるからです。しかし、セキュリティーが重視される環境では、監査ディレクトリをほかのファイルシステム内に入れることは許されません。

監査ファイルシステムを管理するときは、次の要因も考慮する必要があります。

- 各ホストには、少なくとも 1 つのローカル監査ディレクトリを用意する必要があります。このローカルディレクトリは、ホストが監査サーバーと通信できなかった場合の予備ディレクトリとして使用できます。
- 読み取りオプションと書き込みオプション (`rw`) を使用して、監査ディレクトリをマウントしてください。監査ディレクトリを遠隔マウントするときは、`intr` および `noac` オプションも使用してください。
- 監査ファイルシステムを、格納先の監査サーバー上で一覧してください。エクスポートリストには、サイトで監査対象のすべてのシステムが含まれるべきです。

バイナリ監査ファイルの命名規則

各バイナリ監査ファイルは、自己完結したレコードの集合です。ファイル名には、レコードが生成された時間の範囲と、それを生成したシステム名が含まれます。

バイナリ監査ファイル名

完了した監査ファイルには、次の書式の名前が付いています。

start-time.end-time.system

start-time　監査ファイル内の最初の監査レコードが生成された時刻です

end-time　最後のレコードがファイルに書き込まれた時刻です

system　ファイルを生成したシステム名です

監査ファイルがアクティブである場合は、次の書式の名前が付いています。

start-time`.not_terminated.`*system*

`not_terminated` および閉じられた監査ファイルの名前の例については、659 ページの「`not_terminated` 監査ファイルを整理する方法」を参照してください。

バイナリ監査ファイルのタイムスタンプ

auditreduce コマンドは、ファイル名に含まれるタイムスタンプを手掛かりにして、特定期間内のレコードを検索します。1 か月あるいはそれ以上蓄積された監査ファイルがオンライン上に存在する可能性もあるため、これらのタイムスタンプは重要な意味を持ちます。24 時間以内に生成されたレコードをすべてのファイルから検索するとなると、莫大な時間がかかることがあります。

start-time と *end-time* は 1 秒単位のタイムスタンプです。これらのタイムスタンプは、グリニッジ標準時 (GMT) で指定されます。タイムスタンプの書式は、次のように年が 4 桁で、2 桁ずつの月、日、時、分、秒があとに続きます。

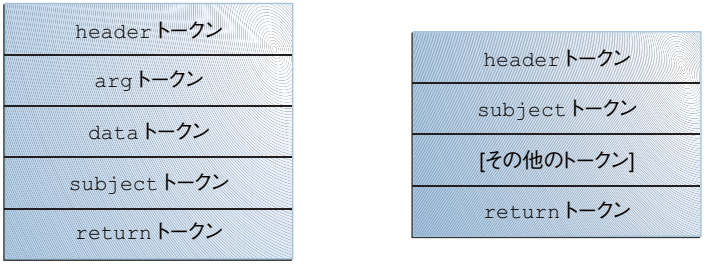
YYYYMMDDHHMMSS

タイムスタンプには GMT が使用されるため、タイムゾーンによるずれがあっても正しい順序でソートされることが保証されます。また、日時を把握しやすいように現在の時間帯に変換する必要があります。監査ファイルを auditreduce コマンドではなく標準ファイルコマンドで操作するときには、この点に注意してください。

監査レコードの構造

監査レコードは、一連の監査トークンです。監査トークンには、ユーザー ID、時刻、日付などのイベント情報が入っています。監査レコードは、header トークンで始まり、オプションの trailer トークンで終わります。ほかの監査トークンには、監査イベントに関連する情報が入っています。次の図は、標準的な監査レコードを示しています。

図 31–3　標準的な監査レコードの構造



監査レコード分析

監査レコード分析には、監査トレールのレコードの事後選択が必要です。次の 2 つの方法のうちのいずれかを使用して、収集されたバイナリデータを解析できます。

- バイナリデータストリームを解析します。データストリームを解析するには、各トークン内のフィールドの順番と各レコード内のトークンの順番を知る必要があります。また、さまざまな監査レコードを知る必要もあります。たとえば、`ioctl()` システムコールは、「Invalid file descriptor」の監査レコードからのさまざまなトークンを含む「Bad file name」の監査レコードを作成します。
  - 各監査トークン内のバイナリデータの順番については、`audit.log(4)` のマニュアルページを参照してください。
  - 監査レコード内のトークンの順番の説明については、`bsmrecord` コマンドを使用してください。`bsmrecord` コマンドによる出力には、さまざまな条件で発生するさまざまな形式が示されます。角括弧 (`[]`) は、監査トークンが省略可能であることを表しています。詳細は、`bsmrecord(1M)` のマニュアルページを参照してください。例については、651 ページの「監査レコードの書式の表示方法」を参照してください。
- `praudit` コマンドを使用します。コマンドのオプションにより、さまざまなテキスト出力が得られます。たとえば、`praudit -x` コマンドを使用すると、スクリプトとブラウザへの入力のための XML が得られます。`praudit` 出力には、単にバイナリデータの解析に使用するためだけのフィールドは含まれません。出力は、バイナリフィールドの順番どおりではありません。また、`praudit` 出力の順番と形式は、Oracle Solaris リリース間で保証されているわけでもありません。

  `praudit` 出力の例については、657 ページの「バイナリ監査ファイルの内容を表示する方法」、および `praudit(1M)` のマニュアルページを参照してください。

  監査トークンごとの `praudit` 出力については、696 ページの「監査トークンの形式」セクションでそれぞれのトークンを参照してください。

監査トークンの形式

各監査トークンにはトークンの種類識別子と、そのあとにトークン固有のデータが続いています。各トークンの種類には固有の形式があります。次の表は、各トークンの名前と簡単な説明の一覧です。廃止されたトークンは、以前の Solaris リリースとの互換性のために維持されています。

表 31–4 Oracle Solaris 監査の 監査トークン

トークン名	説明	詳細
`acl`	アクセス制御リスト (ACL) 情報	698 ページの「`acl` トークン」
`arbitrary`	書式情報と型情報が付いたデータ	698 ページの「`arbitrary` トークン (廃止)」
`arg`	システムコールの引数値	699 ページの「`arg` トークン」
`attribute`	ファイル vnode トークン	700 ページの「`attribute` トークン」

表 31–4 Oracle Solaris 監査の 監査トークン (続き)

トークン名	説明	詳細
cmd	コマンド引数と環境変数	700 ページの「cmd トークン」
exec_args	exec システムコールの引数	701 ページの「exec_args トークン」
exec_env	exec システムコールの環境変数	701 ページの「exec_env トークン」
exit	プログラム終了情報	702 ページの「exit トークン (廃止)」
file	監査ファイル情報	702 ページの「file トークン」
group	プロセスグループ情報	702 ページの「group トークン (廃止)」
groups	プロセスグループ情報	703 ページの「groups トークン」
header	監査レコードの始まりを示します	703 ページの「header トークン」
ip_addr	インターネットアドレス	704 ページの「ip_addr トークン」
ip	IP ヘッダー情報	704 ページの「ip トークン (廃止)」
ipc	System V IPC 情報	704 ページの「ipc トークン」
ipc_perm	System V IPC オブジェクトトークン	705 ページの「ipc_perm トークン」
iport	インターネットポートアドレス	706 ページの「iport トークン」
opaque	構造化されていないデータ (形式が未指定)	706 ページの「opaque トークン (廃止)」
path	パス情報	706 ページの「path トークン」
path_attr	アクセスパス情報	707 ページの「path_attr トークン」
特権	特権設定情報	707 ページの「privilege トークン」
process	プロセスのトークン情報	708 ページの「process トークン」
return	システムコールの状態	710 ページの「return トークン」
sequence	シーケンス番号トークン	710 ページの「sequence トークン」
socket	ソケットの種類とアドレス	711 ページの「socket トークン」
サブジェクト	サブジェクトのトークン (process トークンと同じ書式)	711 ページの「subject トークン」
text	ASCII 文字列	713 ページの「text トークン」
trailer	監査レコードの終わりを示します	714 ページの「trailer トークン」
uauth	承認の使用	714 ページの「uauth トークン」
upriv	特権の使用	714 ページの「upriv トークン」
zonename	ゾーンの名前	715 ページの「zonename トークン」

監査レコードは常に header トークンで始まります。header トークンは、監査トレール内で監査レコードの始まりを示します。ユーザーの動作に起因するイベントの場合、subject と process トークンは、イベントを発生させたプロセスの値を参照します。ユーザーの動作に起因しないイベントの場合、process トークンはシステムを参照します。

acl トークン

acl トークンは、アクセス制御リスト (ACL) に関する情報を記録します。

acl トークンは、次の 4 つの固定長フィールドで構成されます。

- acl トークンであることを特定するトークン ID
- ACL タイプを指定するフィールド
- ACL 値フィールド
- ACL に関連付けるアクセス権を一覧するフィールド

praudit -x コマンドでは、acl トークンのフィールドは次のように表示されます。

```
<acl type="1" value="root" mode="6"/>
```

arbitrary トークン (廃止)

arbitrary トークンは、監査トレール用にデータをカプセル化します。4 つの固定長フィールドと 1 つのデータ配列からなっています。固定長フィールドは次のとおりです。

- arbitrary トークンであることを特定するトークン ID
- 推奨される出力形式フィールド (16 進など)
- カプセル化するデータのサイズを指定する (短い形式など) 項目サイズフィールド
- 後続の項目数を指定するカウントフィールド

トークンの残りの部分は、指定された形式の *count* からなっています。praudit コマンドでは、arbitrary トークンは次のように表示されます。

```
arbitrary,decimal,int,1
42
```

次の表は、出力形式フィールドに指定できる値を示します。

表 31–5 arbitrary トークンの出力形式フィールドの値

値	動作
AUP_BINARY	日付が 2 進形式で出力されます

表 31–5 arbitrary トークンの出力形式フィールドの値 (続き)

値	動作
AUP_OCTAL	日付が 8 進形式で出力されます
AUP_DECIMAL	日付が 10 進形式で出力されます
AUP_HEX	日付が 16 進形式で出力されます
AUP_STRING	日付が文字列で出力されます

次の表は、項目サイズフィールドに指定できる値を示します。

表 31–6 arbitrary トークンの項目サイズフィールドの値

値	動作
AUR_BYTE	データはバイト単位 (1 バイト) です
AUR_SHORT	データは短い形式の単位 (2 バイト) です
AUR_LONG	データは長い形式の単位 (4 バイト) です

arg トークン

arg トークンには、システムコールの引数情報 (システムコールの引数番号、引数の値、およびオプションの説明) が含まれています。このトークンを使用すると、監査レコード内で 32 ビット整数のシステムコール引数を指定できます。

arg トークンには次の 5 つのフィールドがあります。

- arg トークンであることを特定するトークン ID
- トークンが参照するシステムコールの引数の ID
- 引数の値
- テキスト文字列の長さ
- テキスト文字列

praudit -x コマンドでは、arg トークンのフィールドは次のように表示されます。

```
<argument arg-num="2" value="0x0" desc="new file uid"/>
```

attribute トークン

attribute トークンには、ファイル vnode からの情報が含まれています。

attribute トークンには次の 7 つのフィールドがあります。

- attribute トークンであることを特定するトークン ID
- ファイルのアクセスモードと種類
- 所有者のユーザー ID
- 所有者のグループ ID
- ファイルシステム ID
- ノード ID
- ファイルが示すデバイス ID

ファイルシステム ID とデバイス ID の詳細は、statvfs(2) のマニュアルページを参照してください。

attribute トークンには通常、path トークンが付いています。attribute トークンはパスの検索中に生成されます。パス検索エラーが発生すると、必要なファイル情報を取得するための v ノードが利用できません。このため、attribute トークンは監査レコードの一部として組み込まれません。praudit -x コマンドでは、attribute トークンのフィールドは次のように表示されます。

```
<attribute mode="100644" uid="adm" gid="adm" fsid="136" nodeid="2040" device="0"/>
```

cmd トークン

cmd トークンは、コマンドに割り当てられた引数のリストおよび環境変数のリストを記録します。

cmd トークンには次のフィールドがあります。

- cmd トークンであることを特定するトークン ID
- コマンドの引数のカウント
- 引数の一覧
- 次のフィールドの長さ
- 引数の内容
- 環境変数のカウント
- 環境変数の一覧
- 次のフィールドの長さ
- 環境変数の内容

praudit -x コマンドでは、cmd トークンのフィールドは次のように表示されます。次に示すのは、切り詰められた cmd トークンです。行は、表示の都合上、折り返して記載されています。

```
<cmd><arge>WINDOWID=6823679</arge>
<arge>COLORTERM=gnome-terminal</arge>
<arge>...LANG=C</arge>...<arge>HOST=machine1</arge>
<arge>LPDEST=printer1</arge>...</cmd>
```

exec_args トークン

exec_args トークンは、exec() システムコールへの引数を記録します。exec_args トークンには次の 2 つの固定長フィールドがあります。

- exec_args トークンであることを特定するトークン ID
- exec() システムコールに渡す引数の数を表すカウント

このトークンの残りの部分は、*count* 文字列からなっています。praudit -x コマンドでは、exec_args トークンのフィールドは次のように表示されます。

```
<exec_args><arg>/usr/bin/sh</arg><arg>/usr/bin/hostname</arg></exec_args>
```

注 – exec_args トークンは、argv 監査ポリシーオプションが有効なときにだけ出力されます。

---

exec_env トークン

exec_env トークンは、exec() システムコールの現在の環境変数を記録します。exec_env トークンには次の 2 つの固定長フィールドがあります。

- exec_env トークンであることを特定するトークン ID
- exec() システムコールに渡す引数の数を表すカウント

このトークンの残りの部分は、*count* 文字列からなっています。praudit -x コマンドでは、exec_env トークンのフィールドは次のように表示されます。行は、表示の都合上、折り返して記載されています。

```
<exec_env><env>_=/usr/bin/hostname</env>
<env>DTXSERVERLOCATION=local</env><env>SESSIONTYPE=altDt</env>
<env>LANG=C</env><env>SDT_NO_TOOLTALK=1</env><env>SDT_ALT_HELLO=/bin/true</env>
<env>PATH=/usr/bin:/usr/openwin/bin:/usr/ucb</env>
<env>OPENWINHOME=/usr/openwin</env><env>LOGNAME=jdoe</env><env>USER=jdoe</env>
<env>DISPLAY=:0</env><env>SHELL=/bin/csh</env><env>START_SPECKEYSD=no</env>
<env>SDT_ALT_SESSION=/usr/dt/config/Xsession2.jds</env><env>HOME=/home/jdoe</env>
<env>SDT_NO_DTDBCACHE=1</env><env>PWD=/home/jdoe</env><env>TZ=US/Pacific</env>
</exec_env>
```

注 – exec_env トークンは、argv 監査ポリシーオプションが有効なときにだけ出力されます。

exit トークン (廃止)

exit トークンは、プログラムの終了状態を記録します。exit トークンには次のフィールドがあります。

- exit トークンであることを特定するトークン ID
- exit() システムコールに渡されるプログラムの終了状態
- 終了状態を記述するか、システムエラー番号を示す戻り値

praudit コマンドでは、exit トークンは次のように表示されます。

```
exit,Error 0,0
```

file トークン

file トークンは、auditd デーモンによって生成される特殊なトークンです。このトークンは、古い監査ファイルが終了した時点で、新しい監査ファイルの開始と古い監査ファイルの終了をマークします。最初の file トークンは、監査証跡の前のファイルを特定します。最後の file トークンは、監査証跡の次のファイルを特定します。auditd デーモンは、このトークンを含む特殊な監査レコードを構築して、連続する監査ファイルを 1 つの監査トレールに「リンク」します。

praudit -x コマンドでは、file トークンのフィールドは次のように表示されます。このトークンは、監査証跡の次のファイルを特定します。行は、表示の都合上、折り返して記載されています。

```
<file iso8601="2009-04-08 14:18:26.200 -07:00">
/var/audit/machine1/files/20090408211826.not_terminated.machine1</file>
```

group トークン (廃止)

このトークンは、groups トークンに置き換えられています。703 ページの「groups トークン」を参照してください。

groups トークン

groups トークンは group トークンを置き換えます。groups トークンは、プロセスの資格からグループエントリを記録します。

group トークンには次の 2 つの固定長フィールドがあります。

- groups トークンであることを特定するトークン ID
- この監査レコードに含まれるグループ数を表すカウント

このトークンの残りの部分は、*count* グループエントリからなっています。

praudit -x コマンドでは、groups トークンのフィールドは次のように表示されます。

```
<group><gid>staff</gid><gid>other</gid></group>
```

注 – groups トークンは、group 監査ポリシーオプションが有効なときにだけ出力されます。

---

header トークン

header トークンは、監査レコードの開始を示すという意味で、特殊なトークンです。trailer トークンとの組み合わせでレコード内のほかのすべてのトークンを囲む特殊なトークンです。

header トークンには次の 8 つのフィールドがあります。

- header トークンであることを特定するトークン ID
- この監査レコード全体の長さのバイト数。header トークンと trailer トークンを含みます
- この監査レコード構造体のバージョンを特定するバージョン番号
- このレコードが表す監査イベントを特定する監査イベント ID
- この監査イベントの特殊な特性を特定する ID 修飾子

  ID 修飾子フィールドでは、次のフラグが定義されています。

```
0x4000          PAD_NOTATTR                     nonattributable event
0x8000          PAD_FAILURE                     failed audit event
```

- アドレスの種類。IPv4 または IPv6
- マシンのアドレス
- レコードの作成日時

64 ビットシステムでは、header トークンは、32 ビットタイムスタンプではなく 64 ビットタイムスタンプで表示されます。

praudit コマンドでは、header トークンは次のように表示されます。

```
header,69,2,su,,machine1,2009-04-08 13:11:58.209 -07:00
```

praudit -x コマンドでは、header トークンのフィールドは監査レコードの先頭に表示されます。行は、表示の都合上、折り返して記載されています。

```
<record version="2" event="su" host="machine1"
iso8601="2009-04-08 13:11:58.209 -07:00">
```

ip_addr トークン

ip_addr トークンには、インターネットプロトコルアドレスが含まれます。Solaris 8 リリースから、IPv4 形式、IPv6 形式のいずれかでインターネットアドレスを表示できるようになりました。IPv4 アドレスは 4 バイトを使用します。IPv6 アドレスは、1 バイトを使って種類を記述し、さらに 16 バイトを使ってアドレスを記述します。

in_addr トークンには次の 3 つのフィールドがあります。

- in_addr トークンであることを特定するトークン ID
- IP アドレスの種類。IPv4 または IPv6
- IP アドレス

praudit -x コマンドでは、ip_addr トークンの内容は次のように表示されます。

```
<ip_address>machine1</ip_address>
```

ip トークン (廃止)

ip トークンには、インターネットプロトコルのヘッダーのコピーが含まれます。ip トークンには次の 2 つのフィールドがあります。

- ip トークンであることを特定するトークン ID
- IP ヘッダーのコピー (全部で 20 バイト)

praudit コマンドでは、ip トークンは次のように表示されます。

```
ip address,0.0.0.0
```

IP ヘッダー構造は、/usr/include/netinet/ip.h ファイル内で定義されています。

ipc トークン

ipc トークンには、呼び出し元が特定の IPC オブジェクトを識別するために使用する System V IPC メッセージハンドル、セマフォーハンドル、または共有メモリーハンドルが含まれています。

ipc トークンには次の 3 つのフィールドがあります。

- ipc トークンであることを特定するトークン ID
- IPC オブジェクトの形式を指定する形式フィールド
- IPC オブジェクトを識別するハンドル

注 – IPC オブジェクト識別子は、コンテキストに依存しない Oracle Solaris 監査トークンの性質に準拠していません。IPC オブジェクトを一意に識別するグローバルな「名前」はありません。代わりに、IPC オブジェクトはハンドルで識別されます。これらのハンドルは、IPC オブジェクトの動作中にのみ有効です。しかし IPC オブジェクトの識別は問題となりません。System V の IPC メカニズムはあまり使用されず、すべてのメカニズムが同じ監査クラスを共有するからです。

次の表は、IPC オブジェクトの形式フィールドに指定できる値の一覧です。値は /usr/include/bsm/audit.h ファイル内で定義されます。

表 31–7 IPC オブジェクトの形式フィールドの値

名前	値	説明
AU_IPC_MSG	1	IPC メッセージオブジェクト
AU_IPC_SEM	2	IPC セマフォーオブジェクト
AU_IPC_SHM	3	IPC 共有メモリーオブジェクト

praudit -x コマンドでは、ipc トークンのフィールドは次のように表示されます。

```
<IPC ipc-type="shm" ipc-id="15"/>
```

ipc_perm トークン

ipc_perm トークンには、System V IPC アクセス権のコピーが含まれています。このトークンは、IPC 共有メモリーイベント、IPC セマフォーイベント、および IPC メッセージイベントによって生成される監査レコードに追加されます。

ipc_perm トークンには次の 8 つのフィールドがあります。

- ipc_perm トークンであることを特定するトークン ID
- IPC 所有者のユーザー ID
- IPC 所有者のグループ ID
- IPC 作成者のユーザー ID
- IPC 作成者のグループ ID
- IPC のアクセスモード
- IPC のシーケンス番号

- IPC 鍵の値

praudit -x コマンドでは、ipc_perm トークンのフィールドは次のように表示されます。行は、表示の都合上、折り返して記載されています。

```
<IPC_perm uid="jdoe" gid="staff" creator-uid="jdoe"
creator-gid="staff" mode="100600" seq="0" key="0x0"/>
```

値は、IPC オブジェクトに関連付けられた ipc_perm 構造から取り出されます。

iport トークン

iport トークンには、TCP または UDP ポートアドレスが含まれています。

iport トークンには次の 2 つのフィールドがあります。

- iport トークンであることを特定するトークン ID
- TCP または UDP ポートのアドレス

praudit コマンドでは、iport トークンは次のように表示されます。

```
ip port,0xf6d6
```

opaque トークン (廃止)

opaque トークンには、フォーマットされていないデータが一連のバイトとして含まれています。opaque トークンには次の 3 つのフィールドがあります。

- opaque トークンであることを特定するトークン ID
- このデータのバイト数
- バイトデータ配列

praudit コマンドでは、opaque トークンは次のように表示されます。

```
opaque,12,0x4f5041515545204441544100
```

path トークン

path トークンには、オブジェクトのアクセスパス情報が含まれています。

path トークンには次のフィールドがあります。

- path トークンであることを特定するトークン ID
- パスの長さ
- システムの実ルートを基点としたオブジェクトへの絶対パス

praudit コマンドでは、path トークンは 2 番目のフィールドなしで、次のように表示されます。

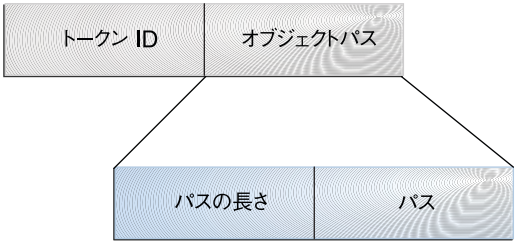
```
path,/etc/security/audit_user
```

praudit -x コマンドでは、path トークンの内容は次のように表示されます。

```
<path>/etc/security/prof_attr</path>
```

次の図に path トークンの形式を示します。

図 31–4 path トークンの形式



path_attr トークン

path_attr トークンには、オブジェクトのアクセスパス情報が含まれています。アクセスパスは、path トークンオブジェクトの下の属性ファイルオブジェクトのシーケンスを指定します。openat() などのシステムコールは、属性ファイルにアクセスします。属性ファイルオブジェクトの詳細については、fsattr(5) のマニュアルページを参照してください。

path_attr トークンには次のフィールドがあります。

- path_attr トークンであることを特定するトークン ID
- 属性ファイルパスのセクション数を表すカウント
- *count* NULL で終わっている文字列

praudit コマンドでは、path_attr トークンは次のように表示されます。

```
path_attr,1,attr_file_name
```

privilege トークン

privilege トークンは、プロセス上での特権の使用を記録します。privilege トークンは、基本セットの特権に対して記録されません。特権が管理者の処理により基本セットから削除された場合、その特権の使用は記録されます。特権の詳細は、199 ページの「特権 (概要)」を参照してください。

privilege トークンには次のフィールドがあります。

- privilege トークンであることを特定するトークン ID
- 次のフィールドの長さ
- 特権セットの名前
- 次のフィールドの長さ
- 特権の一覧

praudit -x コマンドでは、privilege トークンのフィールドは次のように表示されます。行は、表示の都合上、折り返して記載されています。

```
<privilege set-type="Effective">file_chown,file_dac_read,
file_dac_write,net_privaddr,proc_exec,proc_fork,proc_setid</privilege>
```

process トークン

process トークンには、シグナルの受信者など、プロセスに関連付けられたユーザーに関する情報が含まれています。

process トークンには次の 9 つのフィールドがあります。

- process トークンであることを特定するトークン ID
- 監査 ID
- 実効ユーザー ID
- 実効グループ ID
- 実ユーザー ID
- 実グループ ID
- プロセス ID
- 監査セッション ID
- デバイス ID とマシンアドレスで構成される端末 ID

監査 ID、 ユーザー ID、 グループ ID、 プロセス ID、 セッション ID は、短い形式ではなく長い形式です。

注 – セッション ID、 実ユーザー ID、 または実グループ ID の process トークンのフィールドを使用できないことがあります。その場合、値は -1 に設定されます。

端末 ID を含むトークンには、いくつかの種類があります。praudit コマンドは、これらの違いを吸収します。このため、端末 ID を含むすべてのトークンで、端末 ID は同じ方法で処理されます。端末 ID は、IP アドレスとポート番号の組み合わせか、デバイス ID です。モデムに接続されたシリアルポートなどのデバイス ID は、0 である可能性があります。端末 ID には、次の書式があります。

デバイス番号の場合、端末 ID は次のようになります。

- 「32 ビットアプリケーション」 – 4 バイトのデバイス番号、4 バイトは未使用
- 「64 ビットアプリケーション」 – 8 バイトのデバイス番号、4 バイトは未使用

Solaris 8 よりも前のリリースのポート番号の場合、端末 ID は次のようになります。

- 「32 ビットアプリケーション」 – 4 バイトのポート番号、4 バイトの IP アドレス
- 「64 ビットアプリケーション」 – 8 バイトのポート番号、4 バイトの IP アドレス

Solaris 8 以降のリリースのポート番号の場合、端末 ID は次のようになります。

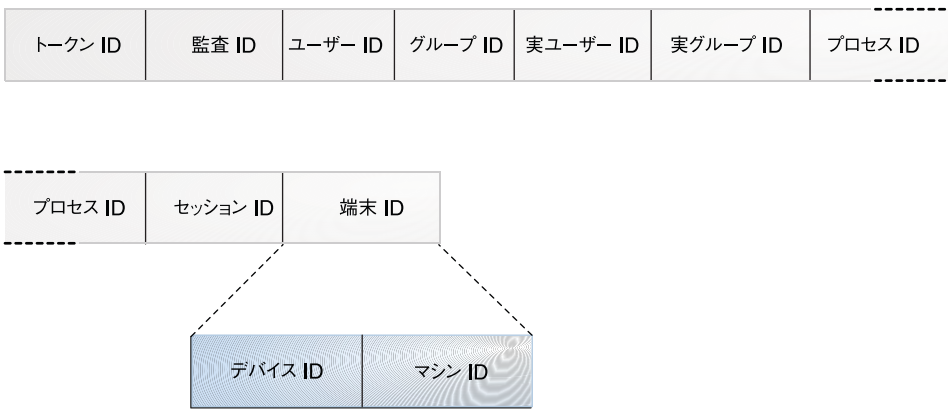
- 「32 ビット IPv4」 – 4 バイトのポート番号、4 バイトの IP タイプ、4 バイトの IP アドレス
- 「32 ビット IPv6」 – 4 バイトのポート番号、4 バイトの IP タイプ、16 バイトの IP アドレス
- 「64 ビット IPv4」 – 8 バイトのポート番号、4 バイトの IP タイプ、4 バイトの IP アドレス
- 「64 ビット IPv6」 – 8 バイトのポート番号、4 バイトの IP タイプ、16 バイトの IP アドレス

`praudit -x` コマンドでは、`process` トークンのフィールドは次のように表示されます。行は、表示の都合上、折り返して記載されています。

```
<process audit-uid="-2" uid="root" gid="root" ruid="root"
rgid="root" pid="9" sid="0" tid="0 0 0.0.0.0"/>
```

次の図に `process` トークンの形式を示します。

図 31–5 process トークンの形式



return トークン

return トークンには、システムコールの戻り状態 (u_error) とプロセスの戻り値 (u_rval1) が含まれています。

return トークンには次の 3 つのフィールドがあります。

- return トークンであることを特定するトークン ID
- システムコールのエラー状態
- システムコールの戻り値

return トークンは、必ずシステムコールに関してカーネルによって生成される監査レコードの一部として返されます。アプリケーションの監査中、このトークンは終了状態とその他の戻り値を返します。

praudit では、システムコールの return トークンは次のように表示されます。

```
return,failure: Operation now in progress,-1
```

praudit -x コマンドでは、return トークンのフィールドは次のように表示されます。

```
<return errval="failure: Operation now in progress" retval="-1/">
```

sequence トークン

sequence トークンには、シーケンス番号が含まれています。シーケンス番号は、監査レコードが監査トレールに組み込まれるたびに 1 ずつ増分します。このトークンはデバッグに使用されます。

sequence トークンには次の 2 つのフィールドがあります。

- sequence トークンであることを特定するトークン ID
- シーケンス番号が含まれる 32 ビットの符号なし長形式フィールド

praudit コマンドでは、sequence トークンのフィールドは次のように表示されます。

```
sequence,1292
```

praudit -x コマンドでは、sequence トークンの内容は次のように表示されます。

```
<sequence seq-num="1292"/>
```

注 – sequence トークンは、seq 監査ポリシーが有効なときにだけ出力されます。

---

socket トークン

socket トークンには、インターネットソケットを記述する情報が含まれています。いくつかのインスタンスで、トークンには次の 4 つのフィールドがあります。

- socket トークンであることを特定するトークン ID
- 参照するソケットの型 (TCP、 UDP、 UNIX) を示すソケット形式フィールド
- ローカルポート
- ローカル IP アドレス

praudit コマンドでは、socket トークンのインスタンスは次のように表示されます。

```
socket,0x0002,0x83b1,localhost
```

ほとんどのインスタンスで、トークンには次の 8 つのフィールドがあります。

- socket トークンであることを特定するトークン ID
- ソケットドメイン
- 参照するソケットの型 (TCP、 UDP、 UNIX) を示すソケット形式フィールド
- ローカルポート
- アドレスの種類。IPv4 または IPv6
- ローカル IP アドレス
- 遠隔ポート
- 遠隔 IP アドレス

Solaris 8 リリースから、IPv4 形式、IPv6 形式のいずれかでインターネットアドレスを表示できるようになりました。IPv4 アドレスは 4 バイトを使用します。IPv6 アドレスは、1 バイトを使って種類を記述し、さらに 16 バイトを使ってアドレスを記述します。

praudit コマンドでは、socket トークンは次のように表示されます。

```
socket,0x0002,0x0002,0x83cf,example1,0x2383,server1.Subdomain.Domain.COM
```

praudit -x コマンドでは、socket トークンのフィールドは次のように表示されます。行は、表示の都合上、折り返して記載されています。

```
<socket sock_domain="0x0002" sock_type="0x0002" lport="0x83cf"
laddr="example1" fport="0x2383" faddr="server1.Subdomain.Domain.COM"/>
```

subject トークン

subject トークンは、ある操作を実行するユーザーまたは実行を試みるユーザーを記述します。形式は process トークンと同じです。

subject トークンには次の 9 つのフィールドがあります。

- subject トークンであることを特定するトークン ID
- 監査 ID
- 実効ユーザー ID
- 実効グループ ID
- 実ユーザー ID
- 実グループ ID
- プロセス ID
- 監査セッション ID
- デバイス ID とマシンの IP アドレスで構成される端末 ID

監査 ID、 ユーザー ID、 グループ ID、 プロセス ID、 セッション ID は、短い形式ではなく長い形式です。

注 – セッション ID、 実ユーザー ID、 または実グループ ID の subject トークンのフィールドを使用できないことがあります。その場合、値は -1 に設定されます。

端末 ID を含むトークンには、いくつかの種類があります。praudit コマンドは、これらの違いを吸収します。このため、端末 ID を含むすべてのトークンで、端末 ID は同じ方法で処理されます。端末 ID は、IP アドレスとポート番号の組み合わせか、デバイス ID です。モデムに接続されたシリアルポートなどのデバイス ID は、0 である可能性があります。端末 ID には、次の書式があります。

デバイス番号の場合、端末 ID は次のようになります。

- 「32 ビットアプリケーション」 – 4 バイトのデバイス番号、4 バイトは未使用
- 「64 ビットアプリケーション」 – 8 バイトのデバイス番号、4 バイトは未使用

Solaris 8 よりも前のリリースのポート番号の場合、端末 ID は次のようになります。

- 「32 ビットアプリケーション」 – 4 バイトのポート番号、4 バイトの IP アドレス
- 「64 ビットアプリケーション」 – 8 バイトのポート番号、4 バイトの IP アドレス

Solaris 8 以降のリリースのポート番号の場合、端末 ID は次のようになります。

- 「32 ビット IPv4」 – 4 バイトのポート番号、4 バイトの IP タイプ、4 バイトの IP アドレス
- 「32 ビット IPv6」 – 4 バイトのポート番号、4 バイトの IP タイプ、16 バイトの IP アドレス
- 「64 ビット IPv4」 – 8 バイトのポート番号、4 バイトの IP タイプ、4 バイトの IP アドレス
- 「64 ビット IPv6」 – 8 バイトのポート番号、4 バイトの IP タイプ、16 バイトの IP アドレス

subjectトークンは、必ずシステムコールに関してカーネルによって生成される監査レコードの一部として返されます。praudit コマンドでは、subject トークンは次のように表示されます。

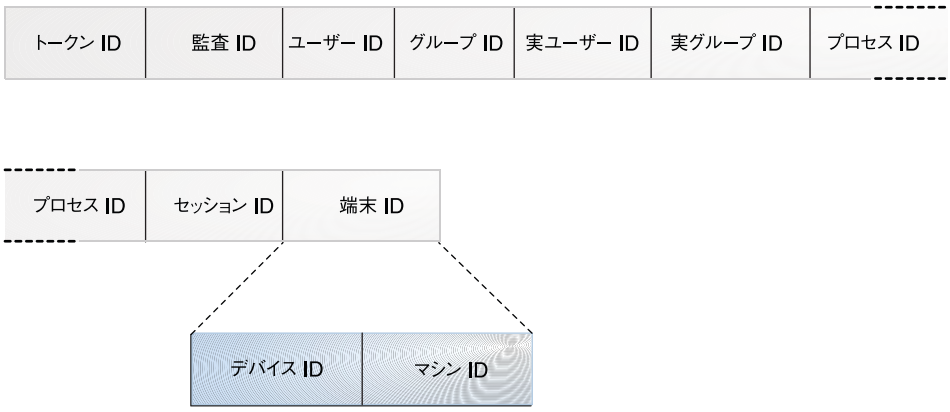
```
subject,jdoe,root,root,root,root,1631,1421584480,8243 65558 machine1
```

praudit -x コマンドでは、subject トークンのフィールドは次のように表示されます。行は、表示の都合上、折り返して記載されています。

```
<subject audit-uid="jdoe" uid="root" gid="root" ruid="root"
rgid="root" pid="1631" sid="1421584480" tid="8243 65558 machine1"/>
```

次の図に subject トークンの形式を示します。

図 31–6 subject トークンの形式



text トークン

text トークンには、テキスト文字列が含まれています。

text トークンには次の 3 つのフィールドがあります。

- text トークンであることを特定するトークン ID
- テキスト文字列の長さ
- テキスト文字列

praudit -x コマンドでは、text トークンの内容は次のように表示されます。

```
<text>booting kernel</text>
```

trailer トークン

header と trailer の2 つのトークンは、監査レコードの終端を区別し、ほかのすべてのトークンを囲むという意味で、特殊なトークンです。header トークンは監査レコードを開始します。trailer トークンは監査レコードを終了します。trailer トークンは省略可能です。trailer トークンは、trail 監査ポリシーオプションが設定されているときにだけ、各レコードの最後のトークンとして追加されます。

trailer トークンを含む監査レコードが生成された場合、 auditreduce コマンドは、trailer がレコードの header を正しくポイントしているかどうかを検証します。また、trailer トークンを使用すると監査トレールを逆方向に検索できます。

trailer トークンには次の 3 つのフィールドがあります。

- trailer トークンであることを特定するトークン ID
- レコードの終了を示すパッド番号
- header トークンと trailerトークンを含む監査レコードの合計文字数

praudit コマンドでは、trailer トークンが次のように表示されます。

```
trailer,136
```

uauth トークン

uauth トークンは、コマンドまたはアクションでの承認の使用を記録します。

uauth トークンには次のフィールドがあります。

- uauth トークンであることを特定するトークン ID
- 次のフィールドのテキストの長さ
- 承認の一覧

praudit コマンドでは、uauth トークンは次のように表示されます。

```
use of authorization,solaris.admin.printer.delete
```

upriv トークン

upriv トークンは、コマンドまたはアクションでの特権の使用を記録します。

praudit -x コマンドでは、upriv トークンのフィールドは次のように表示されます。

```
<use_of_privilege result="successful use of priv">proc_setid</use_of_privilege>
```

zonename トークン

zonename トークンは、監査イベントが発生したゾーンを記録します。文字列「global」は、大域ゾーンで発生した監査イベントを示します。

zonename トークンには次のフィールドがあります。

- zonename トークンであることを特定するトークン ID
- 次のフィールドのテキストの長さ
- ゾーンの名前

praudit -x コマンドでは、zonename トークンの内容は次のように表示されます。

```
<zone name="graphzone"/>
```

用語集

admin 主体
username/admin という形式 (jdoe/admin など) の名前を持つユーザー主体。通常のユーザー主体より多くの特権 (ポリシーの変更など) を持つことができます。主体名とユーザー主体も参照してください。

AES
Advanced Encryption Standard の略。対称 128 ビットブロックのデータ暗号技術。2000 年の 10 月、米国政府は暗号化標準としてこのアルゴリズムの Rijndael 方式を採用しました。ユーザー主体の暗号化に代わる米国政府の標準として、AES が採用されています。

audit policy
どの監査イベントが記録されるかを決定する設定であり、大域の設定とユーザーごとの設定があります。大域の設定は監査サービスに適用され、一般にどのオプション情報を監査トレールに含めるかを決定します。2 つの設定 cnt と ahlt は、監査キューがいっぱいになった時点でのシステムの処理を決定します。たとえば、各監査レコードにシーケンス番号を含めるように監査ポリシーを設定できます。

Blowfish
32 ビットから 448 ビットまでの可変長キーの対称ブロックの暗号化アルゴリズム。その作成者である Bruce Schneier 氏は、鍵を頻繁に変更しないアプリケーションに効果的であると述べています。

DES
Data Encryption Standard。1975 年に開発され、1981 年に ANSI X.3.92 として ANSI で標準化された対称鍵の暗号化方式。DES では 56 ビットの鍵を使用します。

device policy
カーネルレベルでのデバイス保護。デバイスポリシーは、2 つの特権セットとしてデバイスに実装されます。この 1 つはデバイスに対する読み取り権を制御し、もう 1 つはデバイスに対する書き込み権を制御します。ポリシーも参照してください。

Diffie-Hellman プロトコル
公開鍵暗号化としても知られています。1976 年に Diffie 氏と Hellman 氏が開発した非対称暗号鍵協定プロトコルです。このプロトコルを使用すると、セキュリティー保護されていない伝達手段で、事前の秘密情報がなくても 2 人のユーザーが秘密鍵を交換できます。Diffie-Hellman は Kerberos で使用されます。

DSA
デジタル署名アルゴリズム。512 ビットから 4096 ビットまでの可変長キーの公開鍵アルゴリズム。米国政府標準である DSS は最大 1024 ビットです。DSA は入力に SHA1 を使用します。

FQDN
完全指定形式のドメイン名。central.example.com など (単なる denver は FQDN ではない)。

GSS-API
Generic Security Service Application Programming Interface の略。さまざまなモジュールセキュリティーサービス (Kerberos サービスなど) をサポートするネットワーク層。GSS-API は、セキュリティー認証、整合性、およびプライバシサービスを提供します。認証、 整合性、 プライバシも参照してください。

KDC
鍵配布センター (Key Distribution Center)。次の 3 つの Kerberos V5 要素で構成されるマシン。

- 主体と鍵データベース
- 認証サービス
- チケット許可サービス

レルムごとに、1 つのマスター KDC と、1 つ以上のスレーブ KDC を配置する必要があります。

Kerberos
認証サービス、Kerberos サービスが使用するプロトコル、または Kerberos サービスの実装に使用されるコード。

Oracle Solaris Kerberos は、Kerberos V5 認証にほぼ準拠して実装されています。

「Kerberos」と「Kerberos V5」は技術的には異なりますが、Kerberos のマニュアルでは多くの場合、同じ意味で使用されます。

Kerberos (または Cerberus) は、ギリシャ神話において、ハデスの門を警護する 3 つの頭を持つどう猛な番犬のことです。

Kerberos policy
Kerberos サービスでのパスワードの使用方法を管理する一連の規則。ポリシーは、主体のアクセスやチケットのパラメータ (有効期限など) を制限できます。

key
1. 一般には、次に示す 2 種類の主要鍵のどちらか一方です。

- 対称鍵 – 復号化鍵とまったく同じ暗号化鍵。対称鍵はファイルの暗号化に使用されます。
- 非対称鍵または公開鍵 – Diffie-Hellman や RSA などの公開鍵アルゴリズムで使用される鍵。公開鍵には、1 人のユーザーしか知らない非公開鍵、サーバーまたは一般リソースによって使用される公開鍵、およびこれらの 2 つを組み合わせた公開鍵と非公開鍵のペアがあります。非公開鍵は、「秘密鍵」とも呼ばれます。公開鍵は、「共有鍵」や「共通鍵」とも呼ばれます。
- 2. キータブファイルのエントリ (主体名)。キータブファイルも参照してください。

3. Kerberos では暗号化鍵であり、次の 3 種類があります。

- 「非公開鍵」 – 主体と KDC によって共有される暗号化鍵。システムの外部に配布されます。非公開鍵も参照してください。
- 「サービス鍵」 – 非公開鍵と同じ目的で使用されますが、この鍵はサーバーとサービスによって使用されます。サービス鍵も参照してください。
- 「セッション鍵」 – 一時的な暗号化鍵。2 つの主体の間で使用され、その有効期限は 1 つのログインセッションの期間に制限されます。セッション鍵も参照してください。

kvno
鍵バージョン番号。特定の鍵に対して、生成順に付けられたシーケンス番号。もっとも大きい kvno が、最新の鍵を示します。

MAC

1. メッセージ認証コード (MAC)を参照してください。

2.「ラベル付け」とも呼ばれます。政府のセキュリティー用語規定では、MAC は「Mandatory Access Control」の略です。「Top Secret」や「Confidential」というラベルは MAC の例です。MAC と対称をなすものに DAC (Discretionary Access Control) があります。UNIX アクセス権は DAC の 1 例です。

3. ハードウェアにおいては、LAN における一意のシステムアドレス。システムが Ethernet 上に存在する場合は、Ethernet アドレスが MAC に相当します。

MD5

デジタル署名などのメッセージ認証に使用する繰り返し暗号化のハッシュ関数。1991 年に Rivest 氏によって開発されました。

network policies

ネットワークトラフィックを保護するためにネットワークユーティリティーで行われる設定。ネットワークセキュリティーの詳細は、『Solaris のシステム管理 (IP サービス)』のパート IV「IP セキュリティー」を参照してください。

NTP

ネットワークタイムプロトコル (NTP)。デラウェア大学で開発されたソフトウェア。ネットワーク環境で、正確な時間またはネットワーククロックの同期化を管理します。NTP を使用して、Kerberos 環境のクロックスキューを管理できます。「クロックスキュー」も参照してください。

PAM

プラグイン可能認証モジュール (Pluggable Authentication Module)。複数の認証メカニズムを使用できるフレームワーク。認証メカニズムを使用するサービスはコンパイルし直す必要がありません。PAM は、ログイン時に Kerberos セッションを初期化できます。

password policy

パスワードの生成に使用できる暗号化アルゴリズム。パスワードをどれぐらいの頻度で変更すべきか、入力ミスを何回まで認めるかといったセキュリティー上の考慮事項など、パスワードに関連した一般的な事柄を指すこともあります。セキュリティーポリシーにはパスワードが必要です。パスワードポリシーでは、たとえば MD5 アルゴリズムによってパスワードを暗号化することを設定できます。また、パスワードの長さに関連する要件を設定することも可能です。

policy for public key technologies

鍵管理フレームワーク (KMF) におけるポリシーは、証明書の使用を管理します。KMF ポリシーデータベースを使えば、KMF ライブラリによって管理される鍵や証明書の使用に、制約を設けることができます。

policy in the cryptographic framework

Oracle Solaris 暗号化フレームワークでは、ポリシーは既存の暗号化メカニズムの無効化を意味します。無効に設定されたメカニズムは使用できなくなります。暗号化フレームワークでは、ポリシーにより、プロバイダ (DES など) の特定のメカニズム (`CKM_DES_CBC` など) が使用できなくなることがあります。

principal

1. ネットワーク通信に参加する、一意の名前を持つ「クライアントまたはユーザー」あるいは「サーバーまたはサービス」のインスタンス。Kerberos トランザクションでは、主体 (サービス主体とユーザー主体) 間、または主体と KDC の間で対話が行われます。つまり、主体とは、Kerberos がチケットを割り当てることができる一意のエンティティーのことです。主体名、 サービス主体、 ユーザー主体も参照してください。

2. (RPCSEC_GSS API) クライアント主体、 サーバー主体を参照してください。

QOP 保護の質 (Quality of Protection)。整合性サービスまたはプライバシサービスで使用する暗号化アルゴリズムを選択するときに使用されるパラメータの 1 つ。

RBAC Role-Based Access Control の略。役割によるアクセス制御。すべての機能を持つスーパーユーザーの代替アカウント。RBAC を使用すると、組織はスーパーユーザーの機能を分割して、それらを役割と呼ばれる特殊なユーザーアカウントに割り当てることができます。各ユーザーにはそれぞれの責任に応じて役割を割り当てることができます。

RBAC policy コマンドに関連付けられるセキュリティーポリシー。現在、有効なポリシーは `suser` と `solaris` です。`solaris` ポリシーは、特権、承認、および `setuid` セキュリティー属性を認識します。`suser` ポリシーは、`setuid` セキュリティー属性だけを認識します。Oracle Solaris システムとの相互運用が可能な Trusted Solaris システムおよび Trusted Extensions システムでは、特権、`setuid` セキュリティー属性、およびプロセスのラベルを認識する `tsol` ポリシーを使用できます。

RSA デジタル署名と公開鍵暗号化システムを取得するための方法。その開発者である Rivest 氏、Shamir 氏、Adleman 氏によって 1978 年に最初に公開されました。

SEAM Sun Enterprise Authentication Mechanism。ネットワークを介してユーザーの認証を行うシステムの初期バージョンの製品名。マサチューセッツ工科大学 (MIT) で開発された Kerberos V5 技術に準拠します。この製品は、現在 Kerberos サービスと呼ばれています。SEAM は、各種の Solaris リリースには含まれていなかった Kerberos サービスの部分を指します。

Secure Shell セキュリティー保護されていないネットワークを通して、セキュリティー保護された遠隔ログインおよびその他のセキュリティー保護されたネットワークサービスを使用するための特別なプロトコル。

SHA1 セキュリティー保護されたハッシュアルゴリズム。メッセージ要約を作成するために 2^{64} 文字以下の長さを入力するときに操作します。SHA1 アルゴリズムは DSA に入力されます。

stash ファイル stash ファイルには、KDC のマスター鍵を暗号化したコピーが含まれます。サーバーがリブートされると、このマスター鍵を使用して KDC が自動的に認証されてから、`kadmind` プロセスと `krb5kdc` プロセスが起動されます。stash ファイルにはマスター鍵が入っているため、このファイルやこのファイルのバックアップは安全な場所に保管する必要があります。暗号が破られると、この鍵を使用して KDC データベースのアクセスや変更が可能になります。

TGS チケット認可サービス (Ticket-Granting Service)。KDC の構成要素の 1 つ。チケットを発行します。

TGT チケット認可チケット (Ticket-Granting Ticket)。KDC によって発行されるチケット。クライアントは、このチケットを使用して、ほかのサービスのチケットを要求することができます。

アクセス制御リスト **(ACL)**	アクセス制御リスト (ACL) を使用すると、従来の UNIX ファイル保護よりもきめ細かな方法でファイルセキュリティーを確立できます。たとえば、特定のファイルにグループ読み取り権を設定し、そのグループ内の 1 人のメンバーだけにそのファイルへの書き込み権を与えることが可能です。
アプリケーションサーバー	ネットワークアプリケーションサーバーを参照してください。
アルゴリズム	暗号化アルゴリズム。これは、入力を暗号化 (ハッシング) する既成の再帰的な計算手続きです。
暗号化アルゴリズム	アルゴリズムを参照してください。
インスタンス	主体名の 2 番目の部分。インスタンスは、主体の主ノード指定します。サービス主体の場合、インスタンスは必ず指定する必要があります。`host/central.example.com` のように、ホストの完全指定ドメイン名を指定します。ユーザー主体の場合、インスタンスは省略することができます。ただし、`jdoe` と `jdoe/admin` は、一意の主体です。主ノード、主体名、 サービス主体、 ユーザー主体も参照してください。
オーセンティケータ	オーセンティケータは、KDC にチケットを要求するときおよびサーバーにサービスを要求するときに、クライアントから渡されます。オーセンティケータには、クライアントとサーバーだけが知っているセッション鍵を使用して生成された情報が含まれます。オーセンティケータは、最新の識別として検査され、そのトランザクションが安全であることを示します。これをチケットとともに使用すると、ユーザー主体を認証できます。オーセンティケータには、ユーザーの主体名、ユーザーのホストの IP アドレス、タイムスタンプが含まれます。チケットとは異なり、オーセンティケータは一度しか使用できません。通常、サービスへのアクセスが要求されたときに使用されます。オーセンティケータは、そのクライアントとそのサーバーのセッション鍵を使用して暗号化されます。
仮想プライベートネットワーク **(VPN)**	暗号化とトンネルを使用して、セキュリティー保護された通信を提供するネットワーク。公開ネットワークを通してユーザーを接続します。
関係	`kdc.conf` または `krb5.conf` ファイルに定義される構成変数または関係の 1 つ。
監査トレール	すべてのホストから収集した一連の監査ファイル。
監査パーティション	監査ファイルを保持するように設定された、ハードディスク内のパーティション。
監査ファイル	バイナリ形式の監査ログ。監査ファイルは、監査パーティションに別々に保存されます。
キータブファイル	1 つまたは複数の鍵 (主体) が含まれるキーテーブル。ホストまたはサービスとキータブファイルとの関係は、ユーザーとパスワードの関係と似ています。
基本セキュリティーモジュール **(Basic Security Module、 BSM)**	Oracle Solaris の監査サービスとデバイス割り当て。この 2 つの機能によって C2 レベルのセキュリティーが満たされます。

基本セット
: ログイン時にユーザーのプロセスに割り当てられる一連の特権。変更されていないシステムの場合、各ユーザーの初期の継承可能セットはログイン時の基本セットと同じです。

機密性
: プライバシを参照してください。

強化
: ホストが本来抱えるセキュリティー上の脆弱性を解決するためにオペレーティングシステムのデフォルト構成を変更すること。

許可されたセット
: プロセスによって使用できる一連の特権。

クライアント
: 狭義では、`rlogin` を使用するアプリケーションなど、ユーザーの代わりにネットワークサービスを使用するプロセスを指します。サーバー自身が他のサーバーやサービスのクライアントになる場合もあります。

  広義では、a) Kerberos 資格を受け取り、b) サーバーから提供されたサービスを利用するホストを指します。

  広義では、サービスを使用する主体を指します。

クライアント主体
: (RPCSEC_GSS API) RPCSEC_GSS で保護されたネットワークサービスを使用するクライアント (ユーザーまたはアプリケーション)。クライアント主体名は、`rpc_gss_principal_t` 構造体の形式で格納されます。

クロックスキュー
: Kerberos 認証システムに参加しているすべてのホスト上の内部システムクロックに許容できる最大時間。参加しているホスト間でクロックスキューを超過すると、要求が拒否されます。クロックスキューは、`krb5.conf` ファイルに指定できます。

継承可能セット
: プロセスが `exec` の呼び出しを通して継承できる一連の特権。

権利プロファイル
: 権利またはプロファイルとも呼ばれます。RBAC で使用される優先指定の集合。役割またはユーザーに割り当てることができます。権利プロファイルは、承認、特権、セキュリティー属性を指定したコマンド、その他の権利プロファイルから構成できます。

公開鍵の暗号化
: 暗号化方式の 1 つ。各ユーザーが 1 つの公開鍵と 1 つの非公開鍵を所有します。公開鍵の暗号化では、送信者は受信者の公開鍵を使用してメッセージを暗号化し、受信者は非公開鍵を使用してそれを復号化します。Kerberos サービスは非公開鍵システムです。非公開鍵の暗号化も参照してください。

更新可能チケット
: 有効期限の長いチケットは、セキュリティーを低下させることがあるため、「更新可能」チケットに指定することができます。更新可能チケットには 2 つの有効期限があります。 a) チケットの現在のインスタンスの有効期限と、b) 任意のチケットの最長有効期限です。クライアントがチケットの使用を継続するときは、最初の有効期限が切れる前にチケットの有効期限を更新します。たとえば、すべてのチケットの最長有効期限が 10 時間のときに、あるチケットが 1 時間だけ有効だとします。このチケットを保持するクライアントが 1 時間を超えて使用する場合は、チケットの有効期限を更新する必要があります。チケットが最長有効期限に達すると、チケットの有効期限が自動的に切れ、それ以上更新できなくなります。

コンシューマ
: Oracle Solaris 暗号化フレームワークでは、コンシューマはプロバイダが提供する暗号化サービスのユーザー。コンシューマになりえるものとして、アプリケーション、エンドユーザー、カーネル処理などが挙げられます。Kerberos、 IKE、 IPsec などはコンシューマの例です。プロバイダの例は、プロバイダを参照してください。

サーバー
: ネットワーククライアントにリソースを提供する主体。たとえば、システム `central.example.com` に `ssh` する場合、このシステムが `ssh` サービスを提供するサーバーになります。サービス主体も参照してください。

サーバー主体
: (RPCSEC_GSS API) サービスを提供する主体。サーバー主体は、*service@host* という書式で ASCII 文字列として格納されます。クライアント主体も参照してください。

サービス
: 1. ネットワーククライアントに提供されるリソース。多くの場合、複数のサーバーから提供されます。たとえば、マシン `central.example.com` に `rlogin` する場合、このマシンが `rlogin` サービスを提供するサーバーになります。

  2. 認証以外の保護レベルを提供するセキュリティーサービス (整合性またはプライバシ)。整合性とプライバシも参照してください。

サービス鍵
: サービス主体と KDC によって共有される暗号化鍵。システムの外部に配布されます。keyも参照してください。

サービス主体
: 1 つまたは複数のサービスに Kerberos 認証を提供する主体。サービス主体では、一次名はサービス名 (`ftp` など) で、インスタンスはサービスを提供するシステムの完全指定ホスト名になります。ホスト主体、 ユーザー主体も参照してください。

最小化
: サーバーを稼働させる上で必要な最小限のオペレーティングシステムをインストールすること。サーバーの処理に直接影関係がないソフトウェアはすべて、インストールされないか、あるいはインストール後削除されます。

シード
: 乱数生成のスターター (元になる値)。この値から生成が開始されます。このスターターがランダムソースから生じる場合、このシードは「ランダムシード」と呼ばれます。

資格
: チケットと照合セッション鍵を含む情報パッケージ。主体の識別情報を認証するときに使用します。チケットとセッション鍵も参照してください。

資格キャッシュ
: KDC から受信した資格を含む記憶領域。通常はファイルです。

主体名
: 1. 主体の名前。書式は、*primary/instance@REALM*。インスタンス、 主ノード、 レルムも参照してください。

  2. (RPCSEC_GSS API) クライアント主体、 サーバー主体を参照してください。

主ノード
: 主体名の最初の部分。インスタンス、 主体名、 レルムも参照してください。

承認
: 1. Kerberos では、主体がサービスを使用できるかどうか、主体がアクセスできるオブジェクト、各オブジェクトに許可するアクセスの種類を決定するプロセス。

  2. 役割によるアクセス制御 (RBAC) で、役割またはユーザーに割り当てることができるアクセス権。権利プロファイルに埋め込まれていることもあります。承認が与えられていない動作クラスは、セキュリティーポリシーによって拒否されます。

初期チケット	直接発行されるチケット (既存のチケット許可チケットは使用されない)。パスワードを変更するアプリケーションなどの一部のサービスでは、クライアントが非公開鍵を知っていることを確認するために、「初期」と指定されたチケットを要求することができます。初期チケットを使用した検査は、クライアントが最近認証されたことを証明するときに重要になります。チケット許可チケットの場合は、取得してから時間が経過していることがあります。
スーパーユーザーモデル	コンピュータシステムにおける典型的な UNIX セキュリティーモデル。スーパーユーザーモデルでは、管理者は絶対的なシステム制御権を持ちます。一般に、マシン管理のために 1 人のユーザーがスーパーユーザー (root) になり、すべての管理作業を行える状態となります。
スレーブ **KDC**	マスター KDC のコピー。マスター KDC のほとんどの機能を実行できます。各レルムには通常、いくつかのスレーブ KDC (と 1 つのマスター KDC) を配置します。KDC、 マスター KDCも参照してください。
制限セット	プロセスとその子プロセスでどの特権が利用できるかを示す上限。
整合性	ユーザー認証に加えて、暗号チェックサムを使用して、転送されたデータの有効性を提供するセキュリティーサービス。認証、 プライバシも参照してください。
セキュリティーサービス	サービスを参照してください。
セキュリティー属性	RBAC では、セキュリティーポリシーに優先します。セキュリティー属性を使用すると、スーパーユーザー以外のユーザーでも管理コマンドを実行できます。スーパーユーザーモデルでは、setuid プログラムと setgid プログラムがセキュリティー属性です。これらの属性がコマンドで指定されると、そのコマンドがどのようなユーザーによって実行されているかにかかわらず、コマンドは正常に処理されます。特権モデルでは、セキュリティー属性が特権です。特権がコマンドで指定されると、そのコマンドは正常に処理されます。特権モデルは、スーパーユーザーモデルと互換性があります。このため、特権モデルは setuid プログラムと setgid プログラムをセキュリティー属性として認識します。
セキュリティーフレーバ	フレーバを参照してください。
セキュリティーポリシー	ポリシーを参照してください。
セキュリティーメカニズム	メカニズムを参照してください。
セッション鍵	認証サービスまたはチケット認可サービスによって生成される鍵。セッション鍵は、クライアントとサービス間のトランザクションのセキュリティーを保護するために生成されます。セッション鍵の有効期限は、単一のログインセッションに制限されます。keyも参照してください。
ソフトウェアプロバイダ	Oracle Solaris 暗号化フレームワークでは、暗号化サービスを提供するカーネルソフトウェアモジュールまたは PKCS #11 ライブラリ。プロバイダも参照してください。

単一システムイメージ　単一システムイメージは、同じネームサービスを使用する一連の検査対象システムを記述するために、Oracle Solaris 監査で使用されます。これらのシステムは監査レコードを中央の監査サーバーに送信しますが、その監査サーバー上では、それらのレコードがまるで 1 つのシステムからやってきたかのように、レコードの比較を行えます。

遅延チケット　遅延チケットは、作成されても指定された時点まで有効になりません。このようなチケットは、夜遅く実行するバッチジョブなどのために効果的です。そのチケットは盗まれても、バッチジョブが実行されるまで使用できないためです。遅延チケットは、無効チケットとして発行され、a) 開始時刻を過ぎて、b) クライアントが KDC による検査を要求したときに有効になります。遅延チケットは通常、チケット認可チケットの有効期限まで有効です。ただし、その遅延チケットが「更新可能」と指定されている場合、その有効期限は通常、チケット認可チケットの有効期限に設定されます。無効チケット、更新可能チケットも参照してください。

チケット　ユーザーの識別情報をサーバーやサービスに安全に渡すために使用される情報パケット。チケットは、単一クライアントと特定サーバー上の特定サービスだけに有効です。チケットには、サービスの主体名、ユーザーの主体名、ユーザーのホストの IP アドレス、タイムスタンプ、チケットの有効期限を定義する値などが含まれます。チケットは、クライアントとサービスによって使用されるランダムセッション鍵を使用して作成されます。チケットは、作成されてから有効期限まで再使用できます。チケットは、最新のオーセンティケータとともに提示されなければ、クライアントの認証に使用することができません。オーセンティケータ、資格、サービス、セッション鍵も参照してください。

チケットファイル　資格キャッシュを参照してください。

デバイスの割り当て　ユーザーレベルでのデバイス保護。デバイス割り当ては、一度に 1 人のユーザーだけが使用できるようにデバイスを設定する作業です。デバイスデータは、デバイスが再使用される前に消去されます。誰にデバイス割り当てを許可するかは、承認を使用して制限できます。

転送可能チケット　チケットの 1 つ。クライアントが遠隔ホスト上のチケットを要求するときに使用できます。このチケットを使用すれば、遠隔ホスト上で完全な認証プロセスを実行する必要がありません。たとえば、ユーザー `david` がユーザー `jennifer` のマシンで転送可能チケットを取得した場合、david は自分のマシンにログインするときに新しいチケットを取得する必要はありません (再び認証を受けることもない)。プロキシ可能チケットも参照してください。

同期監査イベント　監査イベントの大半を占めます。これらのイベントは、システムのプロセスに関連付けられています。失敗したログインなど、あるプロセスに関連付けられた、ユーザーに起因しないイベントは、同期イベントです。

特権　Oracle Solaris システムにおいてプロセスに対する個々の権利。特権を使用すると、`root` を使用するよりもきめ細かなプロセス制御が可能です。特権の定義と適用はカーネルで行われます。特権の詳細は、`privileges(5)` のマニュアルページを参照してください。

特権エスカレーション　自分に割り当てられたセキュリティー属性 (優先指定を含む) によって許可されるリソースの範囲よりも外側にあるリソースにアクセスすること。その結果、プロセスは未承認のアクセスを実行できることになります。

特権セット
: 一連の特権。各プロセスには、プロセスが特定の特権を使用できるかどうかを判断する 4 セットの特権があります。詳細は、制限セット、 有効セット、 許可されたセット、 および継承可能セットを参照してください。

  基本セットも、ユーザーのログインプロセスに割り当てられる特権セットです。

特権付きアプリケーション
: システム制御に優先するアプリケーション。このようなアプリケーションは、セキュリティー属性 (特定の UID、 GID、 承認、特権など) をチェックします。

特権モデル
: コンピュータシステムにおいてスーパーユーザーモデルより厳密なセキュリティーモデル。特権モデルでは、プロセスの実行に特権が必要です。システムの管理は、管理者が各自のプロセスで与えられている特権に基づいて複数の個別部分に分割できます。特権は、管理者のログインプロセスに割り当てることも、特定のコマンドだけで有効なように割り当てることも可能です。

認証
: 特定の主体の識別情報を検証するプロセス。

ネームサービススコープ
: 特定の役割が操作を許可されている適用範囲。つまり、特定のネームサービス (NIS、 NIS+、 LDAP など) を使用する個々のホストまたはすべてのホストです。スコープは、Solaris 管理コンソールツールボックスに適用されます。

ネットワークアプリケーションサーバー
: ネットワークアプリケーションを提供するサーバー (`ftp` など)。レルムは、複数のネットワークアプリケーションサーバーで構成されます。

ハードウェアプロバイダ
: Oracle Solaris 暗号化フレームワークにおいては、デバイスドライバとそのハードウェアアクセラレータを指します。ハードウェアプロバイダを使用すると、コンピュータシステムから負荷の高い暗号化処理を解放され、その分 CPU リソースをほかの用途に充てることができます。プロバイダも参照してください。

パスフレーズ
: 非公開鍵がパスフレーズユーザーによって作成されたことを検証するために使用されるフレーズ。望ましいパスフレーズは、10 - 30 文字の長さで英数字が混在しており、単純な文や名前を避けたものです。通信の暗号化と復号化を行う非公開鍵の使用を認証するため、パスフレーズの入力を求めるメッセージが表示されます。

非公開鍵
: 各ユーザー (主体) に与えられ、主体のユーザーと KDC だけが知っている鍵。ユーザー主体の場合、鍵はユーザーのパスワードに基づいています。keyも参照してください。

非公開鍵の暗号化
: 非公開鍵の暗号化では、送信者と受信者は同じ暗号化鍵を使用します。公開鍵の暗号化も参照してください。

非同期監査イベント
: 非同期イベントは、システムイベントの内の少数です。これらのイベントは、プロセスに関連付けられていないため、ブロックした後に起動できるプロセスはありません。たとえば、システムの初期起動や PROM の開始および終了のイベントは、非同期イベントです。

秘密鍵
: 非公開鍵を参照してください。

プライバシ
: セキュリティーサービスの 1 つ。送信データを送信前に暗号化します。プライバシには、データの整合性とユーザー認証も含まれます。認証、 整合性、 サービスも参照してください。

フレーバ　従来は、「セキュリティーフレーバ」と「認証フレーバ」は、認証のタイプ (AUTH_UNIX、 AUTH_DES、 AUTH_KERB) を指すフレーバとして、同じ意味を持っていました。RPCSEC_GSS もセキュリティーフレーバですが、これは認証に加えて、整合性とプライバシのサービスも提供します。

プロキシ可能チケット　クライアントに代わってクライアント操作を行うためにサービスによって使用されるチケット。このことを、サービスがクライアントのプロキシとして動作するといいます。サービスは、チケットを使用して、クライアントの識別情報を所有できます。このサービスは、プロキシ可能チケットを使用して、ほかのサービスへのサービスチケットを取得できますが、チケット認可チケットは取得できません。転送可能チケットと異なり、プロキシ可能チケットは単一の操作に対してだけ有効です。転送可能チケットも参照してください。

プロバイダ　Oracle Solaris 暗号化フレームワークでは、コンシューマに提供される暗号化サービス。プロバイダには、PKCS #11 ライブラリ、カーネル暗号化モジュール、ハードウェアアクセラレータなどがあります。プロバイダは暗号化フレームワークに結合 (プラグイン) されるため、プラグインとも呼ばれます。コンシューマの例は、コンシューマを参照してください。

プロファイルシェル　RBAC では、コマンド行から役割 (またはユーザー) が、その役割の権利プロファイルに割り当てられた任意の特権付きアプリケーションを実行できるようにするシェル。プロファイルシェルには、`pfsh`、 `pfcsh`、 および `pfksh` があります。これらはそれぞれ、Bourne シェル (`sh`)、 C シェル (`csh`)、 および Korn シェル (`ksh`) に対応します。

ホスト　ネットワークを通じてアクセス可能なシステム。

ホスト主体　サービス主体のインスタンスの 1 つ (一次名は `host`)。さまざまなネットワークサービス (`ftp`、 `rcp`、 `rlogin` など) を提供するために設定します。`host/central.example.com@EXAMPLE.COM` はホスト主体の例です。サーバー主体も参照してください。

ポリシー　一般には、意思やアクションに影響を与えたり、これらを決定したりする計画や手続き。コンピュータシステムでは、多くの場合セキュリティーポリシーを指します。実際のサイトのセキュリティーポリシーは、処理される情報の重要度や未承認アクセスから情報を保護する手段を定義する規則セットです。たとえば、セキュリティーポリシーで、システムの監査、特権によるデバイスの保護、6 週ごとのパスワード変更といったことを設定できます。

Oracle Solaris OS の特定領域におけるポリシー実装については、audit policy、 policy in the cryptographic framework、 device policy、 Kerberos policy、 password policy、 RBAC policy を参照してください。

マスター **KDC**　各レルムのメイン KDC。Kerberos 管理サーバー `kadmind` と、認証とチケット認可デーモン `krb5kdc` で構成されます。レルムごとに、1 つ以上のマスター KDC を割り当てる必要があります。また、クライアントに認証サービスを提供する複製 (スレーブ) KDC を任意の数だけ割り当てることができます。

無効チケット	まだ使用可能になっていない遅延チケット。無効チケットは、有効になるまでアプリケーションサーバーから拒否されます。これを有効にするには、開始時期が過ぎたあと、TGS 要求で `VALIDATE` フラグをオンにしてクライアントがこのチケットを KDC に提示する必要があります。遅延チケットも参照してください。
メカニズム	1. データの認証や機密性を実現するための暗号化技術を指定するソフトウェアパッケージ。たとえば、 Kerberos V5、 Diffie-Hellman 公開鍵など。 2. Oracle Solaris 暗号化フレームワークにおいては、特定用途のアルゴリズムの実装。たとえば、認証に適用される DES メカニズム (CKM_DES_MAC など) は、暗号化に適用されるメカニズム (CKM_DES_CBC_PAD) とは別です。
メッセージ認証コード **(MAC)**	データの整合性を保証し、データの出所を明らかにするコード。MAC は盗聴行為には対応できません。
メッセージ要約	メッセージ要約は、メッセージから計算されるハッシュ値です。ハッシュ値によってメッセージはほぼ一意に識別されます。要約は、ファイルの整合性を検証するのに便利です。
役割	特権付きアプリケーションを実行するための特別な ID。割り当てられたユーザーだけが引き受けられます。
有効セット	プロセスにおいて現在有効である一連の特権。
ユーザー主体	特定のユーザーに属する主体。ユーザー主体の一次名には、ユーザー名を指定します。オプションのインスタンスには、対応する資格の使用方法を説明する名前を指定します (`jdoe`、 `jdoe/admin` など)。「ユーザーインスタンス」とも呼ばれます。サービス主体も参照してください。
ユーザーに起因しない監査イベント	開始した人を特定できない監査イベント。AUE_BOOT イベントなど。
要約	メッセージ要約を参照してください。
レルム	1. 1 つの Kerberos データベースといくつかの鍵配布センター (KDC) を配置した論理ネットワーク。 2. 主体名の 3 番目の部分。主体名が `jdoe/admin@ENG.EXAMPLE.COM` の場合、レルムは `ENG.EXAMPLE.COM` です。主体名も参照してください。

索引

数字・記号

A

B

C

E

F

G

H

I

J

K

L

O

P

Q

R

S

あ

か

さ

す

つ

な

に

ま

よ

ら

り

れ

ろ

わ