BIND 9.7の新機能を利用した権威DNSサーバの運用

スマート署名

全自動ゾーン署名

DNSSEC for Humans

- BIND 9.7から導入された、DNSSECの設定をより簡単に行う一連の機能
  - **スマート署名**
  - **全自動ゾーン署名**
  - RFC 5011への対応
  - Dynamic Update設定の簡素化
  - DLVの自動設定

スマート署名

スマート署名の利用例（example.jpをNSEC3方式で署名）

```
# ls
example.jp
# dnssec-keygen -3 example.jp
Generating key pair....++++++ ..................++++++
Kexample.jp.+007+31760
# dnssec-keygen -3 -f ksk example.jp
Generating key pair.....................+++ ........+++
Kexample.jp.+007+22740
# dnssec-signzone -3 aabbcc -S example.jp
Fetching ZSK 31760/NSEC3RSASHA1 from key repository.
Fetching KSK 22740/NSEC3RSASHA1 from key repository.
Verifying the zone using the following algorithms: NSEC3RSASHA1.
Zone signing complete:
Algorithm: NSEC3RSASHA1: KSKs: 1 active, 0 stand-by, 0 revoked
                         ZSKs: 1 active, 0 stand-by, 0 revoked
example.jp.signed
# ls
Kexample.jp.+007+22740.key      dsset-example.jp.
Kexample.jp.+007+22740.private  example.jp
Kexample.jp.+007+31760.key      example.jp.signed
Kexample.jp.+007+31760.private
```

スマート署名の利用方法

- 鍵情報を自動的に取り込む
  - ゾーンファイルは通常のゾーンファイル（非DNSSEC）のままでよい
- 主に2つのコマンドを利用
  - 鍵生成には「dnssec-keygen」
  - ゾーン署名には「dnssec-signzone」

コマンドの利用方法（1/2）

- dnssec-keygen
  - デフォルト値
    - アルゴリズム　RSASHA1
      （「-3」を指定すると「NSEC3RSASHA1」）
    - ZSKのbit長　1024 bit
    - KSKのbit長　2048 bit
  - オプション
    - -P：ゾーンへの出力時刻（Publicatijon date）
    - -R：鍵の破棄時刻（Revocation date）
    - -A：署名鍵としての使用開始時刻（Activation date）
    - -I　：署名鍵使用終了時刻（Inactivation date）
    - -D：ゾーンからの削除時刻（Deletion date）
      ※「-P」と「-A」のデフォルトは「now」（現在時刻）

コマンドの利用方法(2/2)

- dnssec-signzone
  - オプション
    - -N：SOAレコード
    - -S：署名するゾーン

注: アルゴリズムを指定する場合、鍵のbit長も指定する
例）RSASHA256での鍵の生成
- dnssec-keygen -a RSASHA256 -b 1024 example.jp
- dnssec-keygen -a RSASHA256 -b 2048 -f ksk example.jp

共通事項など

- 鍵を保存するディレクトリの指定（Key repository）
  - 「-K ディレクトリ名」
- 時刻の指定
  - 絶対時刻
    - 「YYYYMMDD」又は「YYYYMMDDHHMMSS」
  - 相対時刻
    - 「+数字」又は「-数字」
    - ‘y’, ‘mo’, ‘w’, ‘d’, ‘h’, ‘mi’ を指定可能（年、月、週、日、時、分）

コマンドの利用例(1/2)

```
# mkdir keys                                                    ①
# dnssec-keygen -K keys -f ksk example.jp                       ②
Kexample.jp.+005+45154
# dnssec-keygen -K keys -P now -A now -D +31d example.jp        ③
Kexample.jp.+005+20076
# dnssec-keygen -K keys -P now -A +30d -D +61d example.jp       ④
Kexample.jp.+005+45870
# dnssec-signzone -K keys –N unixtime -S example.jp             ⑤
Fetching KSK 45154/RSASHA1 from key repository.
Fetching ZSK 20076/RSASHA1 from key repository.
Fetching ZSK 45870/RSASHA1 from key repository.
Verifying the zone using the following algorithms: RSASHA1.
Zone signing complete:
Algorithm: RSASHA1: KSKs: 1 active, 0 stand-by, 0 revoked
                    ZSKs: 1 active, 1 stand-by, 0 revoked
example.jp.signed
# ls -F                                                         ⑥
dsset-example.jp.       example.jp.signed
example.jp              keys/
```

コマンドの利用例（2/2）

① 鍵用のディレクトリ（keys）を作成
② KSKを作成
③ 最初に使うZSKを作成
  – すぐにゾーンに出力、すぐに署名に使用、31日後に削除
④ 2番目に使うZSKを作成
  – すぐにゾーンに出力、30日後に署名に使用、60日後に削除
⑤ ゾーンへの署名
  – KSKは1個、ZSKは1個が署名用1個が事前公開用
    ※この例ではNSEC方式を採用し、SOAのシリアルはunixtimeを使っている
⑥ 鍵はkeysディレクトリ内にある

署名と鍵更新の自動化

- cron等を利用する
  - 定期的に dnssec-keygen で -P, -A, -I, -D を適正に設定したZSKを作成
  - 定期的に dnssec-signzone -S で再署名

  ⇒ 鍵更新、再署名の自動化が可能になる
- KSKについても同様の処理が可能
  - 但しDSの更新には親ゾーンとのやり取りが必要なため、完全な自動化は難しい

  注意：
  dnssec-keygenで-Aを指定する場合、必ず-Pも指定する
  ⇒ BIND 9.7.2-P3時点での不具合

全自動ゾーン署名

全自動ゾーン署名

- namedがゾーンへの署名、鍵更新を行う
  - dnssec-signzoneは利用しない
  - 鍵にはスマート署名の場合と同様、日付情報を設定する
- ゾーン毎に次のいずれかを設定する
  - auto-dnssec allow; 署名等はrndcコマンド使って で別途制御する（定期的な再署名は行われる）
  - auto-dnssec maintain; 鍵ファイルに記録されている日付情報に基づいて完全に自動化する

全自動ゾーン署名設定の例

```
options {
    ....
    directory          "/var/named";
    session-keyfile    "/var/named/session.key";
    ....
};

zone {
    type               master;
    file               "master/example.jp";
    key-directory      "master/keys";
    update-policy      local;
    auto-dnssec        maintain;
};
```

全自動ゾーン署名設定(1/2)

- session-keyfile
  - ダイナミックアップデートのための鍵ファイルの指定
  - 指定しない場合コンパイル時のデフォルトが適用される
- key-directory
  - スマート署名の-Kで指定するディレクトリと同じもの
- update-policy
  - ダイナミックアップデートのポリシーの設定で、ここでは単純な local を設定
  - 必要に応じて他のポリシーも設定できる

全自動ゾーン署名設定(2/2)

- ゾーンファイルは、非DNSSECのものでよい
- rndcコマンドが正しく動作するよう設定する
- ゾーンファイル、ゾーンファイルのあるディレクトリなどは、namedプロセスの権限で書き換え可能なパーミッションに設定
  - 必要に応じてnamedがファイルを作成したり、書き換えたりするため

auto-dnssec maintain;

- named 起動後、鍵ディレクトリ内の鍵ファイルの日付に応じてゾーンに署名を行う
  - dnssec-signzone -Sと同様の動作となる
- KSKとZSKをNSEC3用に設定しても、デフォルトはNSECとなる（DNSSECとしては問題無い）
- NSEC3で運用するための二つの方法
  - ダイナミックアップデート（nsupdateコマンド）を使い、NSEC3PARAMレコードを追加する
    ⇒ 追加直後にNSEC3方式に切り替わる
  - 予めゾーンファイルにNSEC3PARAMレコードを登録
    ⇒ 起動直後の最初の署名でNSEC3方式になる

nsupdateコマンド

- 稼働中のゾーンデータに、動的にRRの追加削除（ダイナミックアップデート）を行うコマンド
- NSEC3PARAM RRを追加する例

```
# nsupdate -k /var/named/session.key -l
> update add example.jp 0 nsec3param 1 0 5 AABBCCDD
> send
> quit
#
```

  - -l（エル）でローカルのnamedを指定
- 更新情報はジャーナルファイルに記録される
  - この例では”/var/named/master/example.jp**.jnl**”になる

全自動ゾーン署名時のゾーン情報の変更

- ゾーンファイルはnamedが直接管理するため、単純には編集できない
- 解1: ダイナミックアップデートを利用する
  - nsupdateコマンドを利用して、RRの追加、削除、変更を行う
- 解2: 一時的にnamedがゾーンファイルを更新するのを停止させ通常通り編集し、namedのゾーンファイルの更新を再開する

全自動ゾーン署名のゾーンファイルの編集

- 一時的にゾーンファイルの更新を停止する

```
# rndc freeze example.jp
```

  - この時点でnamedが保持しているゾーン情報がすべてexample.jpのゾーンファイルに反映される
- 編集する

```
# vi example.jp
```

  - RRSIGなどが追加されているが気にしなくて良い
- ゾーンファイルの更新を再開する

```
# rndc thaw example.jp
```

  - 再開した時点でnamedがゾーンデータを読み込み、再署名が行われる

鍵の追加と署名

- ZSKの追加(KSKも同様)
  - dnssec-keygenで日付情報を適正に指定したZSKを生成し鍵のディレクトリに用意する
  - 鍵ファイルの追加後rndcコマンドで鍵情報の変更をnamedに通知

```
# rndc loadkeys example.jp
```

  - dnssec-settimeで鍵の日付情報を変更した場合も同様の処理が必要
- DNSSEC運用に必須の定期的な再署名は自動的に行われる
  - なんらかの理由により署名を行いたい場合

```
# rndc sign example.jp
```

全自動ゾーン署名の注意点

- DS RRの作成
  - dnssec-signzoneを利用しないため、DS RRは、KSKの鍵ファイルから作成する必要がある

```
# dnssec-dsfromkey Kexample.jp.+005+35251.key >
  dsset-example.jp.
```

- 長期間運用を続けると鍵ファイルが増える
  - 使い終わった鍵ファイルを削除する仕組みを、別途用意する必要がある
    ⇒ スマート署名の場合も同じ

まとめ

運用面から見たスマート署名

- ゾーンファイルの変更履歴はとりやすい
- 署名完了後のゾーンファイルをnamedに読み込ませるため、ある程度確実な運用ができる
- dnssec-signzoneには差分署名の機能が無いため、大きなゾーンファイルに対しては、ゾーン変更時の署名の負荷が大きい
  - 但しDNSSECの運用では定期的な再署名が必要であり、再署名時は全署名となるため、スマート署名だから問題になるものではない

運用面から見た全自動ゾーン署名

- 比較的運用が単純化できる
- ダイナミックアップデートを使う場合は差分情報のみ署名されるため、ゾーン情報変更時の負荷が軽い
- ゾーンファイル内にRRSIGやNSEC等が自動的に追加されるため、ゾーンファイルの更新履歴を記録しにくい
- 運用実績があまり無いためトラブルシュートに対する不安が残る