



Federal University of Pernambuco
Informatics Center
Doctorate in Computer Science

**“An Orchestration Approach for Unwanted
Internet Traffic Identification”**

Eduardo Luzeiro Feitosa

Recife, August 2010.



Federal University of Pernambuco
Informatics Center
Doctorate in Computer Science

**“An Orchestration Approach for Unwanted
Internet Traffic Identification”**

Eduardo Luzeiro Feitosa

This thesis has been submitted to the Informatics Center of the Federal University of Pernambuco as a partial requirement to obtain the degree of Doctor in Computer Science.

Supervisor: Prof. Dr. Djamel Fawzi Hadj Sadok
Co-Supervisor: Prof. Dr. Eduardo James Pereira Souto

Recife, August 2010.

Feitosa, Eduardo Luzeiro

An orchestration approach for unwanted internet traffic identification / Eduardo Luzeiro Feitosa. - Recife: O Autor, 2010.

xiii, 172 folhas : il., fig., tab.

Tese (doutorado) Universidade Federal de Pernambuco. CIn. Ciência da Computação, 2010.

Inclui bibliografia.

1. Redes de computadores. 2. Segurança da informação. 3. Medições de tráfego.
I. Título.

004.6

CDD (22. ed.)

MEI2010 – 0146

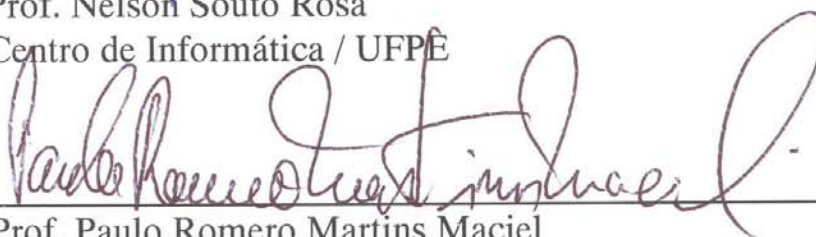
Tese de Doutorado apresentada por **Eduardo Luzeiro Feitosa** a Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**An Orchestration Approach for Unwanted Internet Traffic Identification**", elaborada sob a orientação do **Prof. Djamel Fawzi Hadj Sadok** e aprovada pela Banca Examinadora formada pelos professores:



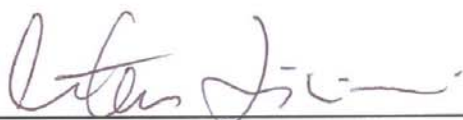
Prof. Paulo Roberto Freire Cunha
Centro de Informática / UFPE



Prof. Nelson Souto Rosa
Centro de Informática / UFPE



Prof. Paulo Romero Martins Maciel
Centro de Informática / UFPE



Prof. Arthur Ziviani
Laboratório Nacional de Computação Científica



Prof. Carlos Alberto Kamienski
Universidade Federal do ABC

Visto e permitida a impressão.
Recife, 30 de agosto de 2010.



Prof. NELSON SOUTO ROSA

Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

To my parents, wife and sons

Acknowledgments

The fulfillment of this Thesis would not have been possible without the contribution of a large number of persons. The first persons I am deeply indebted to are my wife Livia Soraya and my children's Gabriel, Bruna and Luísa. While they did not contribute to this Thesis directly, but I would like to thank them for their support and love. I would also like to thank my parents, Clarice and José Ribamar, for their important support.

I owe a big thank-you to my Thesis advisor, Djamel Sadok, who offered the opportunity to join his research team a few years ago. I must say that at that time, I was not really imagining the kind of experience I was about to embark on. I appreciated very much Djamel's very pragmatic approach to networking. Djamel is also an inexhaustible source of networking references. Through his enthusiasm and unlimited support (time, ideas and experience), he helped me to complete this Thesis. In addition to that, this work contains the fruits of many and lengthy discussions regarding the present contributions with Djamel.

My sincere thanks also go to Professor Judith Kelner for giving me as well as to my family a great deal of support when we moved to Recife. She was instrumental in my participation in the GPRT group. I learned a lot from her valuable experience. I am also thankful for the excellent example she has provided as a successful researcher and Professor.

The third person I would like to thank is my Thesis co-advisor Eduardo Souto. Souto has been a great friend from Manaus, where we worked together in the University's Data Processing Center (CPD). He was key to my decision for choosing Recife as the place to do my Doctorate studies. I learned a lot from the references he pointed me to and from the many discussions we had. We collaborated on a lot of problems, especially about the design of the OADS Miner.

I am grateful to the external members of my Thesis examination committee, namely professors, Artur Ziviani, Carlos Alberto Kamienski, Nelson Rosa and Paulo Maciel, and must thank them warmly for accepting to be part of this work. I would especially like to thank Carlos Alberto Kamienski and Nelson Rosa for their detailed comments and recommendations when defending the proposal for this Thesis, more than a year ago.

Let me also thank the members of GPRT for their direct collaboration and participation in this Thesis: Bruno Lins, Rodrigo Melo, Leo Vilaça, Thiago Rodrigues, and Fernando Rodrigues. Bruno Lins helped me to develop the Alert Pre-Processor tool. In addition to this collaboration, I am also obliged to thank him for using his tool for Dempster-Shafer analysis from C++ to Java in this Thesis. Rodrigo Melo helped implementing almost all the experiments presented in this Thesis. Leo Vilaça and I collaborated on the study of the frequent episode analysis and the joint development of the Alert Analyzer tool. Thiago Rodrigues and I collaborated on the characterization of the OASD Miner (ARAPONGA) tool and we had several discussions on how to turn this more efficient with the extraction of security information from the Web. Fernando Rodrigues helped me a lot improve the writing of many parts of this Thesis.

Other members of GPRT that also deserve my thanks due to their contributions with different points of the study are: Rafael Aschoff, Guthemberg Silvestre, Patricia Endo, Luis Eduardo Oliveira, and Josias Junior. I would also acknowledge the work done by Manuela Melo e Nadia Silva and their contributions to the pleasant ambiance that reigns in GPRT.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of this thesis.

Eduardo Luzeiro Feitosa

August 30, 2010

Resumo

Um breve exame do atual tráfego Internet mostra uma mistura de serviços conhecidos e desconhecidos, novas e antigas aplicações, tráfego legítimo e ilegítimo, dados solicitados e não solicitados, tráfego altamente relevante ou simplesmente indesejado. Entre esses, o tráfego Internet não desejado tem se tornado cada vez mais prejudicial para o desempenho e a disponibilidade de serviços, tornando escasso os recursos das redes. Tipicamente, este tipo de tráfego é representado por spam, *phishing*, ataques de negação de serviço (DoS e DDoS), vírus e *worms*, má configuração de recursos e serviços, entre outras fontes.

Apesar dos diferentes esforços, isolados e/ou coordenados, o tráfego Internet não desejado continua a crescer. Primeiramente, porque representa uma vasta gama de aplicações de usuários, dados e informações com diferentes objetivos. Segundo, devido a ineficácia das atuais soluções em identificar e reduzir este tipo de tráfego. Por último, uma definição clara do que é não desejado tráfego precisa ser feita.

A fim de solucionar estes problemas e motivado pelo nível atingido pelo tráfego não desejado, esta tese apresenta:

1. Um estudo sobre o universo do tráfego Internet não desejado, apresentado definições, discussões sobre contexto e classificação e uma série de existentes e potenciais soluções.
2. Uma metodologia para identificar tráfego não desejado baseada em orquestração. OADS (*Orchestration Anomaly Detection System*) é uma plataforma única para a identificação de tráfego não desejado que permite um gerenciamento cooperativo e integrado de métodos, ferramentas e soluções voltadas a identificação de tráfego não desejado.
3. O projeto e implementação de soluções modulares integráveis a metodologia proposta. A primeira delas é um sistema de suporte a recuperação de informações na Web (WIRSS), chamado OADS Miner ou simplesmente ARAPONGA, cuja função é reunir informações de segurança sobre vulnerabilidades, ataques, intrusões e anomalias de tráfego disponíveis na Web, indexá-las eficientemente e fornecer uma máquina de busca focada neste tipo de informação. A segunda, chamada *Alert Pre-Processor*, é um esquema que utilize uma técnica de *cluster* para receber múltiplas fontes de alertas, agregá-los e extrair aqueles mais relevantes, permitindo correlações e possivelmente a percepção das estratégias usadas em ataques. A terceira e última é um mecanismo de correlação e fusão de alertas, *FER Analyzer*, que utilize a técnica de descoberta de episódios frequentes (FED) para encontrar sequências de alertas usadas para confirmar ataques e possivelmente prever futuros eventos.

De modo a avaliar a proposta e suas implementações, uma série de experimentos foram conduzidos com o objetivo de comprovar a eficácia e precisão das soluções.

Palavras-Chave: Tráfego Internet não Desejado, Orquestração, Correlação de Alertas, Descoberta de Episódios Frequentes, WIRSS.

Abstract

A brief examination of the current Internet traffic shows a varying mix of known and unknown services, legacy and new applications, legitimate and illegitimate traffic, solicited and unsolicited data, highly relevant and unwanted traffic. Among these, unwanted Internet traffic is increasingly becoming harmful to network performance and service availability, often taking up processing and scarce network resources. Typically, unwanted traffic is represented, in general, by spoofing activities, spam, phishing, DoS and DDoS, virus and worms, misconfiguration, or among other sources.

Nonetheless, there are many isolated and coordinated efforts to deal with this issue, unwanted Internet traffic continues to grow. First, because basically unwanted traffic represents a wide range of user applications, network data and harmful information with different objectives for its existence. Secondly, the inefficiency of the current solutions to identify, reduce, and stop unwanted traffic is notorious. The increase in Internet link bandwidth and service mix makes the timely detection of unwanted traffic an interminable task that does not scale easily as such links increase in capacity. Lastly, a clear definition of what is unwanted traffic remains to be elaborated.

In order to address these problems and motivated by the current alarming situation that unwanted traffic has reached, this thesis presents:

4. A study of unwanted Internet traffic universe, presenting definitions, discussing about context and classifications, and a series of existing and potential solutions.
5. An approach to identify unwanted traffic based on orchestration defined as OADS. OADS (Orchestration Anomaly Detection System) is a single-platform for unwanted traffic identification management to allow an integrated management of all cooperative methods, tools and events for unwanted traffic identification.
6. The design and implementation of three tools. The first one is a Web Information Retrieval Support System (WIRSS), called OADS Miner or ARAPONGA, gathering security information about vulnerabilities, attacks, intrusions and traffic anomalies available on the Web, indexing them efficiently, and providing a focused search engine. The second one is an Alert Pre-Processor tool, which employs a cluster approach to receive multi-source alerts, aggregates them and extracts the most relevant, allowing their correlation and possibly perception of their attack strategy. The third and last one is an alert correlation and fusion tool, OADS FER Analyzer, which employs a Frequent Episodes Discovery (FED) technique to discover sequences of alerts applied to confirm attacks and predict future events.

In order to evaluate this proposal and its implementations, a set of experimentation were conducted, aiming to prove the efficacy and accuracy of tools.

Keywords: Unwanted Internet Traffic, Orchestration, Alert Correlation, Frequent Episodes Discovery, WIRSS

Contents

PART I - BACKGROUND	1
CHAPTER 1.....	2
INTRODUCTION	
1.1. OBJECTIVES AND CONTRIBUTIONS	4
1.2. ROAD MAP	5
1.3. BIBLIOGRAPHIC NOTES	6
CHAPTER 2.....	8
UNWANTED INTERNET TRAFFIC.....	
2.1. DEFINITION	8
2.1.1. Context problem	9
2.2. CLASSIFICATION	9
2.3. UNWANTED TRAFFIC: WHO IS GUILTY?	12
2.3.1. Internet Design Principles	12
2.3.2. The root of all evil	15
2.4. RECENT EXAMPLES OF UNWANTED TRAFFIC.....	16
2.4.1. Internet infrastructure attacks	16
2.4.2. SPAM	19
2.4.3. Malicious code.....	21
2.4.4. Social networks	22
2.4.5. Recreational traffic	23
2.5. CHAPTER SUMMARY	24
CHAPTER 3.....	26
APPROACHES AGAINST UNWANTED TRAFFIC.....	
3.1. TRADITIONAL SOLUTIONS.....	26
3.1.1. Filtering	26
3.1.2. Intrusion Detection System	27

3.1.3.	Anti-something software	27
3.1.4.	Honeypot	28
3.2.	PROMISING SOLUTIONS	28
3.2.1.	Advanced Filtering	28
3.2.2.	IP address space investigation	29
3.3.	COLLABORATIVE SOLUTIONS.....	29
3.3.1.	Alert Normalization	30
3.3.2.	Alert Aggregation	31
3.3.3.	Alert Correlation.....	31
3.3.4.	False Alert Reduction	33
3.3.5.	Alert Prioritization.....	34
3.3.6.	Alert Prediction	35
3.4.	TRAFFIC ANALYSIS	36
3.4.1.	Non-Gaussian and Long Memory Statistical Characterizations for Internet Traffic with Anomalies.....	36
3.4.2.	Extracting Hidden Anomalies using Sketch and Non Gaussian Multiresolution Statistical Detection Procedures	37
3.4.3.	A Novel Approach for Anomaly Detection over High-Speed Networks .	39
3.4.4.	Anomaly Detection of Network Traffic based on Wavelet Packet	41
3.4.5.	Profiling Internet backbone Traffic: Behavior Models and Applications	42
3.4.6.	Mining Anomalies Using Traffic Feature Distributions	43
3.4.7.	Discussion of evaluated traffic analysis approaches	47
3.5.	CHAPTER SUMMARY	48
PART II - OADS APPROACH AND TOOLS.....		49
CHAPTER 4.....		50
ORCHESTRATION ANOMALY DETECTION SYSTEM (OADS)		
4.1.	OADS OVERVIEW	51
4.2.	OADS ARCHITECTURE.....	51
4.2.1.	Anomaly Detectors	52
4.2.2.	Alert Pre-Processor	52
4.2.3.	Alert Analyzer	53
4.2.4.	Decision Service	54
4.2.5.	OADS Miner	55

4.3.	OADS CONTRIBUTIONS	56
4.4.	CHAPTER SUMMARY	56
CHAPTER 5.....		58
OADS ALERT PRE-PROCESSOR		
5.1.	BACKGROUND.....	59
5.1.1.	Entropy and Relative Uncertainty	59
5.1.2.	Extracting Significant Clusters.....	60
5.2.	ALERT PRE-PROCESSOR ARCHITECTURE	61
5.3.	IMPLEMENTATION.....	62
5.3.1.	Alert Handler Module.....	62
5.3.2.	Aggregation Module.....	64
5.4.	PERFORMANCE EVALUATION.....	65
5.4.1.	Benchmarking.....	65
5.5.	STRESS TEST	67
5.5.1.	Alert files AF1 and AF2	67
5.5.2.	DARPA 2000 Dataset.....	69
5.6.	CHAPTER SUMMARY	75
CHAPTER 6.....		76
OADS FER ANALYZER.....		
6.1.	FREQUENT EPISODES RULES (FER)	77
6.1.1.	Basic Concepts	77
6.1.2.	Related Work.....	80
6.2.	FER ANALYZER: DESIGN AND IMPLEMENTATION.....	80
6.2.1.	Alert Handler module	81
6.2.2.	Frequent Episode Analysis module	82
6.2.3.	Implementation.....	86
6.3.	EVALUATION	86
6.3.1.	Performance overview	86
6.3.2.	Episode Rules	91
6.3.3.	Real Traffic Analysis.....	96
6.4.	CHAPTER SUMMARY	97
CHAPTER 7.....		99

OADS MINER: CODENAME ARAPONGA	
7.1. BACKGROUND AND RELATED WORK.....	99
7.2. ARAPONGA OVERVIEW	102
7.2.1. Templates	104
7.2.2. Architecture	106
7.2.3. Components Interaction.....	107
7.3. IMPLEMENTATION.....	108
7.3.1. Preliminary Questions	108
7.3.2. Components	111
7.4. EVALUATION AND INITIAL RESULTS	116
7.4.1. Performance Metrics.....	116
7.4.2. Evaluation Methodology	117
7.4.3. Experiment Results.....	118
7.5. CHAPTER SUMMARY	121
PART III - RESULTS	123
CHAPTER 8.....	124
OADS IMPLEMENTATION AND PERFORMANCE EVALUATION.....	
8.1. OADS IMPLEMENTATION	124
8.1.1. ADS-Fusion	125
8.1.2. Decision Service	127
8.1.3. Enforcement Actions	128
8.2. HEURISTIC FOR ORCHESTRATION.....	128
8.3. TESTBED ENVIRONMENT	131
8.3.1. Malicious traffic generation.....	133
8.4. ORCHESTRATING ANALYSIS	133
8.4.1. Scan UPnP	133
8.4.2. DNS cache poisoning	135
8.4.3. SMTP Flood	139
8.4.4. Slowloris.....	141
8.4.5. Multi-step Attack.....	146
8.4.6. Experimenting with real traces	150
8.5. CHAPTER DISCUSSION	151

CHAPTER 9.....	152
CONCLUSIONS.....	
9.1. SUMMARY OF CONTRIBUTIONS	152
9.2. LESSONS LEARNED	153
9.2.1. Content Selection vs. Crawler Tool (OADS Miner)	153
9.2.2. Detectors and IDMEF.....	154
9.2.3. Real traffic traces.....	155
9.3. FUTURE DIRECTIONS	155
9.3.1. Distributed support and cooperation	155
9.3.2. Secure and trusty relationship	155
9.3.3. Studies on distribution of the OADS orchestration.....	156
9.3.4. Improvements in Orchestration algorithm.....	156
9.3.5. Design and implementation of an inter-domain advertisement service .	156
9.3.6. Information bases	156
9.4. FINAL REMARKS	157
REFERENCES	158

List of Figures

2.1: HTTP “hourglass model”	13
2.2: Spam email offering a DDoS attack service.....	19
2.3: SPIM example.	21
3.1: Sketches generation process	38
3.2: Anomaly detection stage	39
3.3: Profiling Methodology	43
3.4: ChkModel Sock and Flow Schema	45
3.5: ChkModel Architecture	45
4.1: E-mail example of the fight against unwanted traffic	50
4.2: OADS architecture view.....	52
4.3: Snort and Prelude nomenclature problem.	53
4.4: Example of attack evaluation by two anomaly detectors.	54
4.5: OADS Miner overview.	55
5.1: Functional diagram of OADS Alert Pre-Processor.	62
5.2: Translation example of ChkModel output to IDMEF format	63
5.3: Data structure of ATable and CTable.....	64
5.4: CPU load of the Aggregation module from alert file 2 (AF2)	66
5.5: Memory usage of the Aggregation module from alert file 2 (AF2)	66
5.6: Significant clusters extracted from class of attack dimension in AF2	68
5.7: Significant clusters extracted from destination IP address dimension in AF2.....	69
5.8: Significant clusters extracted from source IP address dimension in AF2.....	69
5.9: Class of attack frequency distribution in LLDOS 1.0 inside scenario	70
5.10: Destination IP frequency distribution in LLDOS 1.0 inside scenario.....	71
5.11: Source IP clusters dispersion in LLDOS 1.0 inside scenario	72
5.12: Class of attack frequency distribution in LLDOS 1.0 outside scenario	72
5.13: Destination IP frequency distribution in LLDOS 1.0. outside scenario.....	73
5.14: Source IP frequency distribution in LLDOS 1.0 outside scenario	73
5.15: Class of attack frequency distribution in LLDOS 2.0.2 inside scenario	74
5.16: Destination IP frequency distribution in LLDOS 2.0.2 inside scenario.....	74

5.17: Class of attack frequency distribution in LLDOS 2.0.2 outside scenario	74
5.18: Destination IP frequency distribution in LLDOS 2.0.2 outside scenario.....	75
6.1: A Graphical Representation of the Sequence of Events s	77
6.2: Time Window within Sequence s.....	78
6.3: X, Y and Z represent serial, parallel and non-serial non-parallel episodes.....	78
6.4: Functional diagram of FER Analyzer.....	81
6.5: Event sequence representation	82
6.6: Normal rules and reduced rule for a sadmind request in LLDOS 1.0 Outside	86
6.7: Number of frequent episodes as a function of window size, with frequency threshold of 0.002, for LLDOS 1.0 scenarios.....	88
6.8: Number of frequent episodes as a function of window size, with frequency threshold 0.002, for LLDOS 2.0.2 scenarios.....	90
6.9: Rule space generated from LLDOS 1.0 Inside.....	91
6.10: Rule space generated from LLDOS 1.0 Outside	91
6.11: Rule space generated from LLDOS 2.0.2 Inside.....	92
6.12: Rule space generated from LLDOS 2.0.2 Outside	92
6.13: The effects of pruning for LLDOS 1.0 Inside, with various confidence thresholds, with frequency threshold 0.005 and window size 20	93
6.14: The effects of pruning for LLDOS 1.0 Outside, with various confidence thresholds, with frequency threshold 0.005 and window size 20.....	93
6.15: The effects of pruning for LLDOS 2.0.2 Inside, with various confidence thresholds, with frequency threshold 0.005 and window size 20.....	94
6.16: The effects of pruning for LLDOS 2.0.2 Outside, with various confidence thresholds, with frequency threshold 0.005 and window size 20.....	94
6.17: Sample of the episode rules generated for LLDOS 1.0 Inside (confidence thresholds 0.8, frequency threshold 0.002 and window size 20).....	95
6.18: Translation of an episode rule to IPTable rules.....	95
6.19: Episode rules generated for LLDOS 2.0.2 Inside (confidence thresholds 0.8, frequency threshold 0.002 and window size 20)	96
7.1: A partial multi-level knowledge structure for vulnerability by vendor.....	103
7.2: Differences among search spaces	104
7.3: Secunia Web site Template	105
7.4: Architectural and workflow diagram of ARAPONGA	107
7.5: Example of Vulnerability summary query by Microsoft term	115

7.6: ARAPONGA's GUI Interface.....	116
7.7: Comparison ARAPONGA's GUI Interface	118
7.8: Brazilian ASes related to malicious activities	120
7.9: Frequency distribution in Atlas Web site	121
7.10: A timeline of Microsoft Internet Explorer vulnerabilities.....	121
8.1: OADS implementation architecture	124
8.2: Mapping classifications and enforcement actions	128
8.3: Full OADS Testbed Topology	132
8.4: UPnP alerts time line	133
8.5: Possible IPTables rules for UPnP alerts	134
8.6: Summary_vulnerabilty for SCAN UPnP alert	135
8.7: DNS cache poisoning scenario	136
8.8: DNS cache poisoning attack without defense	136
8.9 Possible IPTables rules for DNS cache poisoning alerts.....	137
8.10: DNS cache poisoning attack with OADS decision and actions	138
8.11: SPAM attack without defense (collected from Firewall/Gateway).....	139
8.12: Possible IPTables rules for SMTP flood attack.....	140
8.13: SPAM attack with defense	141
8.14: Typical slowloris HTTP request.....	142
8.15: Slowloris attack without defense on Web server 2	143
8.16: Slowloris attack without defense on Web server 2	145
8.17: Blaster worm testbed scenario.....	147
8.18: Specific Snort rule to detect Blaster worm simulation.....	147

List of Tables

3.1: A subjective comparison of various anomaly detection techniques.....	47
5.1: Characteristic of alert files AF1 and AF2.....	66
5.2: CPU load and memory usage of the Aggregation module.....	66
5.3: Characteristic of alert files AF1 and AF2.....	67
5.4: Class of Attack relative uncertainty in AF2.....	68
5.5: Class of attack relative uncertainty in LLDOS 1.0 inside scenario.....	71
5.6: Destination IP relative uncertainty in LLDOS 1.0 inside scenario.....	71
6.1: Example of event types, event names and their attributes.....	81
6.2: Performance for LLDOS 1.0 inside scenario.....	87
6.3: Performance for LLDOS 1.0 outside scenario.....	87
6.4: Performance for LLDOS 2.0.2 inside scenario.....	89
6.5: Performance for LLDOS 2.0.2 outside scenario.....	89
6.6: GPRT Laboratory Frequency Episode Results.....	97
7.1: Secunia Advisories template.....	105
7.2: Comparison among various Web sites.....	109
7.3: Heritrix and Nutch comparison.....	110
7.4: Features of Lucene and Jericho.....	112
7.5: Query types of ARAPONGA Interface.....	113
7.6: Base evaluation.....	117
7.7: Comparative study of search results from the query.....	119
8.1: Representation of information sent by detectors to Decision Service.....	127
8.2: FER parameters for orchestration algorithm.....	130
8.3: Distribution of detectors in OADS testbed.....	132
8.4: Calculated bpa's of Web server 1.....	144
8.5: Calculated bpa's of Web server 2.....	144
8.6: Dempster combination for attack in Web server 2.....	146
8.7: Example of event types and event names for Blaster worm scenario.....	148
8.8: Performance for Blaster worm scenario.....	149

Part I
Background

Chapter 1

Introduction

A brief examination of current Internet traffic shows a varying mix of known and unknown services, legacy and new applications, legitimate and illegitimate traffic, solicited and unsolicited data, highly relevant and unwanted traffic. Among these, unwanted Internet traffic is increasingly becoming harmful to network performance and service availability, often taking up scarce precious network and processing resources. Typically, unwanted traffic is generated by backscatter¹ from spoofing activities, unsolicited electronic messages (spam), phishing² and pharming³ attempts, denial of service attacks (DoS and its distributed form - DDoS), virus and worms spreading, misconfiguration, among other sources.

Unwanted Internet traffic can be considered an Internet plague, which consequences are reflected in financial losses around the world. Statistics provided by the Computer Security Institute (CSI) [1] and public security agencies such as the American FBI indicate that the financial losses occasioned by network attacks, intrusions and anomalies reached approximately US\$ 252 million in last three years [2] [3][4]. The Gartner Inc. [5] reveals that financial fraud hit 7.5% of Americans in 2008. The Radicati Group [6] estimated global losses equivalent to US\$ 198 billion related to spam messages in 2007. In addition, they predicted that the number of spam messages would reach 79% of the volume of global e-mail messages in 2010. Recent studies prove that the proliferation of this traffic is so fast that 3G networks [7] are beginning to feel its negative effects.

In Brazil, despite the lack of an accurate figure showing the level of financial losses, the statistics provided by the CSIRT (Computer Security Incident Response Team) reveal an alarming increase of the number of incidents. The CERT.br (Computer Emergency Response Team Brazil) [8] reported 222,528 security incidents in 2008, an increase of 39% in relation to 2007, where almost 62.3% were fraud attempts. It also recorded 108,242 notifications of breach of copyright via distribution on P2P networks. The CAIS (Centro de Atendimento a Incidentes de Segurança) [9] recorded more than 35.000 security incidents.

In spite of the old presence of unwanted traffic in the Internet, only recently some serious isolated and coordinated efforts are being taken to deal with this problem and the losses it has been causing. This is witnessed, among other things, by the large number of workshops and conferences dedicated to the exchange of experience and tools in the combat of this phenomenon. Examples of such meetings include SRUTI⁴ (Steps to Reducing Unwanted Traffic on the Internet) and the setup of an Internet

¹ Backscatter is the traffic received from victims that are responding to denial of service attacks.

² Phishing is a form of electronic fraud, characterized by attempts to acquire sensitive information, such as passwords and credit card numbers, to be getting as a trustworthy person or a company sending a communication electronics officer as a mail or an instant message.

³ Pharming is a technique that uses kidnapping or "contamination" of the DNS (Domain Name Service) to redirect users to a fake domain. It can also redirect users to sites not authentic through proxies controlled by phishers, which can be used to track and intercept the typing.

⁴ <http://www.usenix.org/events/sruti05/>

Architecture Board (IAB) working group on unwanted traffic that resulted in the RFC 4849 [10], in addition to flagship communication and networking conferences such as the ACM SIGCOMM⁵.

Despite these separate efforts, this type of traffic continues to grow. First, because basically unwanted traffic represents a wide range of user applications, network data and harmful information with different objectives for its existence. Some of it may be pure nuisance such as spam messages, other consists of bulky multimedia content result from technological trends in applications, networks and users' habits, including P2P file sharing (e.g., Emule, Bit torrents), video sharing (e.g., Justin.tv⁶, Joost⁷, YouTube⁸), and recreational traffic (e.g., MP3 downloads, instant messaging, Skype, MSN), and finally one finds specially designed intrusive traffic targeting networking resources and service availability such as worms, viruses, and denial of service attacks.

Secondly, the inefficiency of the current solutions to identify, reduce, and stop unwanted traffic is notorious as well described in recent works [10][11][12]. The increase in Internet link bandwidth and service mix makes the timely detection of unwanted traffic a boring task that does not scale easily as such links increase in capacity. Typically, the existing solutions trigger alarms often after some damage was already caused. More active strategies need to be put in place to speed the detection and response up. Nowadays, while some users are recovering from given problems from unwanted traffic, others are next to be subjected to it.

Lastly, a clear definition of what is unwanted traffic remains to be made. Someone's unnecessary or even harmful traffic may be seen as someone else's normal service. Recreational applications such as online games, instant messages, P2P applications, VoIP and video services, and emerging social networks can be considered normal activities by given ISPs, while being inappropriate in most enterprise networks. Since it is hard to get consensus over this, a single and flexible solution capable to accommodate all concerns becomes more and more difficult.

To sum up, this thesis is motivated by the current alarming situation that unwanted traffic has reached. It becomes clear that this type of traffic stands as one of the key security problems and one that needs urgent identification and mitigation although it is still not trivial how to do so. Issues as types, sources and goals need to be carefully studied and answered so that effective actions can be undertaken to mitigate the effect of the unwanted traffic.

This thesis shares the view that:

- Limiting the line of defense to peripheral mechanisms provides a small picture of a wide scenario.
- The use of highly specialized systems to combat specific problems can achieve little benefit and these are inefficient in handling other types of unwanted traffic including new ones.
- The distributed and collaborative solutions are primordial to get an early interception and filtering out of suspicious traffic and to ensure a limited damage by such type of traffic.

⁵ <http://www.sigcomm.org/>

⁶ <http://www.justin.tv>

⁷ <http://www.joost.com>

⁸ <http://www.youtube.com>

1.1. Objectives and Contributions

Actually, strategies for dealing with unwanted traffic are based on three steps: (i) to first gain knowledge of the different types and sources of unwanted traffic, (ii) to assess their impact and the effectiveness of existing solutions against these, and (iii) to develop and test effective new countermeasures against unwanted traffic.

The main goal of this thesis is to achieve these steps focusing on the study, definition and description of the problem of unwanted Internet traffic identification and consequently putting forward a solution that is able to accurately identify this traffic.

Overall, there are five contributions on this thesis:

1. The **study of unwanted Internet traffic universe**. The work begins by introducing the reader into the world of unwanted Internet traffic. Definitions, classifications and gives some examples designed to explain the growth of this traffic (Chapter 2). Next, many solutions to deal with unwanted traffic are also presented (Chapter 3).
2. An **approach to identify unwanted traffic based on orchestration** (Chapter 4). The proposal of a single-platform for unwanted traffic identification management to allow an integrated management of all cooperative methods, tools and events for unwanted traffic identification is detailed next. Denominated OADS (Orchestration-oriented Anomaly Detection System), it specifies a framework capable to receive multiples inputs (alerts) from different anomaly detectors, evaluates them and decides of any likely existence of some type of traffic anomaly.
3. The **design and implementation of the Alert Pre-Processor tool** (Chapter 5). This tool employs a cluster approach to receive multi-source alerts, aggregates them and extracts the most relevant, allowing their correlation and possibly perception of their attack strategy. To sum up, this tool has the potential to reduce the bandwidth and computational load at the (centralized) server(s), decreasing the false negative rate and prioritizing the most relevant alerts.
4. The **design and implementation of an alert correlation and fusion tool**, OADS FER Analyzer (Chapter 6). More specifically, this tool relies on the gathering and correlation of incoming alerts, aiming to discover sequences (or frequent episodes) and predict future alerts and consequently possible targets of anomalies. To achieve this, it adopts episode frequency analysis, an adaptive technique that observes and develops knowledge, in the form of probabilistic rules, to generate relationships among events (alerts) that anticipate and make up a given attack.
5. The **design and implementation of a Web Information Retrieval Support System (WIRSS) tool**, OADS Miner (Chapter 7). This tool gathers security information about vulnerabilities, attacks, intrusions, and traffic anomalies available on the Web. It has two main applications. The former collects data from distinct and reliable Internet sources and indexes them efficiently, excluding irrelevant information. The latter provides a focused search engine that allows ISP network operators, IT managers and researchers to better understand of the causes, effects and trends involving attacks and anomalies on the Internet.

In order to evaluate the ideas behind this Thesis and its implementations, a series of experimentation were conducted, aiming to prove the efficacy and accuracy of the adopted solutions. The obtained results can be used to give insights and ideas towards building future robust network architectures that may avoid some of the pitfalls and unpredicted shortcomings of protocol design leading to unwanted traffic domination of the Internet.

1.2. Road map

This Thesis is organized in three parts. The first part provides the background required to understanding this work. It consists of two surveys: one on unwanted Internet traffic universe and other one relates to solutions suggested in dealing with it. Readers who are familiar with unwanted Internet traffic definitions and known solutions against it can skip this part. The second part describes the design aspects of the proposed approach for unwanted traffic identification. The implementation of each component is showed in individual chapters, describing some related works and the artifacts employed for their deployment. The third part shows the evaluation results of the current unified proposal. The implemented tools are tested in different scenarios and using both controlled and injected real Internet traffic.

Part I - Background

In **Chapter 2**, background notions required to understand the Thesis are provided. First, an overview of unwanted Internet traffic is introduced. Some formal definitions are presented and a new one is provided. Next, existing classifications of unwanted traffic are presented and a new one, related to legitimate traffic, is formalized. Furthermore, the most relevant points, aspects and shortcomings on the Internet architecture design are discussed to establish a relationship with the current level of unwanted Internet traffic found today. The existence of an underground economy, commonly related with attacks and the proliferation of this traffic, is also discussed. Last, the most recent unwanted traffic types are presented, including Internet infrastructure attacks, popular “social activities” like spam, phishing, and P2P applications; malicious codes, and uncommon traffic as encapsulated traffic.

Chapter 3 discusses unwanted traffic solutions and the combat to do away with it. First, some current mechanisms and solutions are presented and their failures with regard to unwanted traffic are discussed. Second, some promising and relevant solutions are described. In this section, a survey of approaches to traffic analysis, together with special emphasis on traffic behavior analysis and its applicability to identify, characterize, and detect unwanted Internet traffic, are presented. Last, collaborative solutions are described, including requirements to develop this type of solution and many related works.

Part II – OADS Approach and Tools

Chapter 4 explains the Thesis vision for what is the next step in direction of the automatic and quick detection and limitation of unwanted Internet traffic. The proposal of the Orchestration oriented Anomaly Detection System (OADS) is presented. The idea behind this approach is to harmonize a range of components (anomaly detectors, information bases, alert handlers, analyzers, and decision service) via an orchestration engine, emulating the interaction among different and distinct elements and increasing the accuracy of a diagnosis towards an event. The OADS approach facilitates the management of unwanted traffic identification, by providing means to integrate

(collaboration) different anomaly detectors and consequently increasing the network security level.

In **Chapter 5**, the design and implementation of OADS Alert Pre-Processor tool is presented. It focuses on solving the problem of the large number of overwhelming alerts generated by intrusion and anomaly detectors whenever abnormal or suspicious activities are detected. Since inspecting and investigating all reported alerts manually is a difficult, error-prone, and time-consuming task, the concepts of entropy and relative uncertainty are applied to developed a simple tool capable to automatically aggregate large volume of multi-sources alerts (without specific prior knowledge about them) and next extracting only the most significant ones. This module is able to deduce simple scenario attacks and take immediate actions.

In **Chapter 6**, the design and implementation of OADS FER Analyzer tool is presented. This chapter evaluates the benefits of the use of frequent episodes analysis to correlate and predict alerts with a high confidence level. Therefore, the implementation of this tool is presented and some tests are made to confirm its efficiency in the presence of anomalous traffic. Making use of a more representative input (only relevant alerts from OADS Alert Pre-Processor), it is proved that the tool is capable to correlate these alerts, allowing a better representation of attack scenarios and taking more effective countermeasures.

In **Chapter 7**, the design and implementation of an OADS Internet Miner tool is presented. The focus of interest is to help ISP network operators and IT managers in enlisting the help specialized Web sites containing vulnerabilities reports and Internet traffic statistics in an attempt to keep up to date with current security threats and incidents, minimize their impacts and speed up both detection and mitigation phases. While making use of information retrieval concepts such Web-based Information Retrieval Support System (WIRSS) and Search Support Engine (SSE), a module focused on gathering information about Internet security events (OADS Miner) is built into the unified solution.

Part III – Results

Chapter 8 describes the implantation and evaluation of the OADS approach. First, the orchestration heuristic is presented and explained. Next, relevant points of the implementation (and not previously discussed) are described to provide a better knowledge of the adopted OADS approach. Therefore, five experiments, representing and emulating different but real scenarios, are conducted in order to test the OADS prototype.

Chapter 9 presents the Thesis conclusion. It reviews and summarizes the Thesis contributions, discusses some lessons learned and points towards interesting future works.

1.3. Bibliographic Notes

Most of the work presented in this Thesis appears in previously published or currently submitted conference proceedings and journals. The list of related publications is shown hereafter:

- L. E. Oliveira, R. Aschoff, B. Lins, **E. L. Feitosa**, D. Sadok. Avaliação de Proteção contra Ataques de Negação de Serviço Distribuídos (DDoS)

utilizando Lista de IPs Confiáveis. In *Anais of VII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg 2007)*. Rio de Janeiro, Brasil: SBC, 2007.

- **E. L. Feitosa**, L. E. Oliveira, B. Lins, A. Carvalho Junior, R. Amorim, D. Sadok, U. Carmo. “Security Information Architecture for Automation and Control Networks”. In *Anais of VIII Brazilian Symposium on Information Security and Computing Systems (SBSeg’08)*. Porto Alegre, Brasil: SBC, pp. 17-30, 2008.
- **E. L. Feitosa**, E. Souto, and D. Sadok, “Tráfego Internet não Desejado: Conceitos, Caracterização e Soluções,” in *Text-Book of Mini courses of the VIII of the Brazilian Symposium on Information Security and Computing Systems (SBSeg’08)*. Porto Alegre, Brasil: SBC, 2008, ch. 3.
- D. Sadok, E. J. Souto, **E. L. Feitosa**, J. Kelner, L. Westberg. “RIP - A Robust IP Access Architecture”. *Computers & Security*, vol. 38, pp. 359-380, 2009.
- B. Lins, **E. L. Feitosa**, D. Sadok. “Aplicando a Teoria da Evidência na Detecção de Anomalias”. In *Anais of XXVII Brazilian Symposium of Computer Networks and Distributed Systems (SBRC’09)*. Recife, Brasil: SBC, pp. 583-596, 2009.
- T. G. Rodrigues, **E. L. Feitosa**, D. Sadok, J. Kelner. “Uma Ferramenta de Suporte a Recuperação de Informação na Web focada em Vulnerabilidades e Anomalias Internet”. *Submitted to the 10th Brazilian Symposium on Information Security and Computer Systems (SBSEG 2010)*.
- L. Vilaça, **E. L. Feitosa**, D. Sadok, J. Kelner. “Aplicando a Frequência de Episódios na Correlação de Alertas”. *Submitted to the 10th Brazilian Symposium on Information Security and Computer Systems (SBSEG 2010)*.
- **E. L. Feitosa**, T. G. Rodrigues, E. Souto, D. Sadok. “A Web Information Retrieval Support System for Internet Anomalies and Vulnerabilities”. *Submitted to the 11th International Conference on Web Information System Engineering (WISE 2010)*.
- B. Lins, **E. L. Feitosa**, D. Sadok. “Um Esquema para Agregação e Extração de Alertas para Soluções Colaborativas”. *Submitted to the 10th Brazilian Symposium on Information Security and Computer Systems (SBSEG 2010)*.
- **E. L. Feitosa**, L. Vilaça, D. Sadok, J. Kelner. “OADS Alert Pre-Processor: a tool for alert correlation and prediction”. *Submitted to the International Journal on Multi-Sensor, Multi-Source Information Fusion (Elsevier)*.
- **E. L. Feitosa**, E. J. Souto, D. Sadok. “Orchestration oriented Anomaly Detection System”. *Submitted to the Computers & Security Journal (Elsevier)*.

Chapter 2

Unwanted Internet Traffic

Not all unwanted Internet traffic should be seen as harmful. All involved parties (Internet Service Providers - ISP, Telecoms, enterprises, end users, for example) have different views about what is unwanted traffic or not, and, consequently, the way that they react to it is also similarly different.

This chapter defines unwanted Internet traffic according to some works and presents a new definition that includes both legitimate and illegitimate traffic. The problem involving the context is also discussed and illustrated using some examples. Then, a formal classification and a new one based on common traffic and application features is presented. Next, some possible reasons that could explain the recent increase of this type of traffic are enumerated. Architectural Internet design and choices are analyzed and a relationship with unwanted traffic generation is established. Finally, a discussion of the main types or sources of this traffic is made. The main or more known types are explained and examples of results are described.

2.1. Definition

Initially introduced in the beginning of the 80's, the term "unwanted traffic" was always associated with some incidents like viruses, worms, intrusions and attacks. Some of those became famous such as the Internet worm [13] and DDoS attack to eBay, Amazon, and CNN.com [14].

Only recently unwanted traffic has been used to define any Internet unsolicited, non-productive, not desirable, and illegitimate traffic. Pang *et al.* [15] define unwanted traffic as a non-productive traffic composed by malicious (flooding backscatter, scans for vulnerabilities, worms) or benign (misconfiguration) traffic. They term this as background radiation traffic and include backscatter traffic related with DoS and DDoS attacks response, scan activities, spam, and exploits traffic. They also refer to unwanted traffic as "up to no good" traffic. Soto [16] adds that unwanted traffic also can be generated by traffic corrupted by noise or interference on network transmission lines. Indeed there were many occasions where misconfigured routers or faulty networking equipment would send huge amounts of traffic towards networks where it was not needed. Xu *et al.* [17] characterize unwanted traffic as malicious or unproductive traffic that attempts to compromise vulnerable hosts, propagate malware, spread spam or deny the use of valuable services. Other existing nomenclatures are: background traffic [10], abnormal traffic [15] and "junk" traffic [18].

Despite the previous definitions comprehend almost all aspects related to the generation and impact of unwanted traffic, a recent and important factor is not considered: financial gains. For this reason, this thesis defines unwanted traffic as: *any not requested and unwanted network traffic, which its unique purpose or outcome is consuming network and computing resources, wasting communication, processing and storage time and money of the users or the owner of the resources while often generating profitability for hackers in some form* [19].

2.1.1. Context problem

In spite of these definitions, there is no consensus about what is unwanted traffic. Commonly, such “concept” is relative and dependent of the context in which it is located or applied, and of the used application. For example, China treats the traffic generated by SkypeOut application [20], a service that allows Skype voice over IP users to access the worldwide public switched telephone (PSTN) lines, as illegal because it affects the profit of the government owned Telecommunication Company.

Following the same logic, ISPs, Telecoms, and public and private enterprises began to limit the use of Peer-to-Peer (P2P) file sharing and video sharing (YouTube, for example). They argue that the generated traffic of these applications is potentially a breach for distributing malicious code such as spreading viruses, worms, spywares, and bots to their customers. In addition, it breaches copyright protection, consumes unnecessary bandwidth, and wastes work-time of employees.

This same view has been given to recreational applications (online games) and activities such as emerging social networks (Orkut⁹, MySpace¹⁰, Facebook¹¹, for example) and relationship applications such as IRC (Internet Relay Chat), instant messages (MSN¹² and Google Talk¹³, for example). The most recent case involves the Comcast Company, which in October 2007 was secretly discovered degrading several popular peer-to-peer applications, including BitTorrent [21].

2.2. Classification

Typically, the strategy for dealing with unwanted traffic is to gain knowledge about it and then establish deterministic ways for detecting it in the traffic mix. For this reason, it is necessary to categorize and classify unwanted traffic in terms of its nature (root causes, common types, targets and effects).

The first formal unwanted traffic taxonomy was specified at an IAB (Internet Architecture Board) workshop [10]. It proposes a classification of the deliberately created unwanted traffic in enterprise networks into three categories: Nuisance, Malicious, and Unknown.

- **Nuisance**, as the name says, covers the background traffic that clogs bandwidth and resources like computing power and storage. Typical examples include Spam and P2P file sharing since this kind of traffic normally carries malware or lures the user to access unreliable links. Regarding to P2P traffic, there is the cumbersome issue of the infringement of copyright. Beyond Spam and P2P, this category can also include DoS and DDoS attacks. Denial of Service attacks, as their name suggests, remove temporarily access to a service by bombarding this with service requests. Although one of the most used forms of attacks, they are often quickly detected and cause only loss of service time which may sometimes reflect on enterprise revenues. Often the loss is limited to service unavailability, seen as a nuisance at best. DDoS attacks usually generate an unusual traffic

⁹ <http://www.orkut.com>

¹⁰ <http://www.myspace.com>

¹¹ <http://www.facebook.com>

¹² <http://www.msn.com>

¹³ <http://www.google.com/talk>

profile where many hosts target a single server in a very short duration by sending a very high number of service requests. This traffic profile is used in the detection and identification of such attacks.

- **Malicious** represents the traffic responsible for spreading malware including viruses, worms, spywares, and others. An important fact is that after incidents caused by malicious traffic are detected, the solution demands specific tools, skills and time. The high level of losses that this class of unwanted traffic requires a fast and efficient response and costly constant software updating. In addition, it is normally specific to targeting operating systems, router and other software vulnerabilities. To complicate matters, there are even kits available in the Internet to teach any user how to create new versions of viruses and worms.
- **Unknown** involves all traffic that even when belonging to the above categories it could not be identified as such (malicious traffic encrypted or merged with legitimate traffic, for example) or those that nobody knows anything about their intentions or sources. Quiet worms like Storm [22] are another example. They open backdoors on hosts and stay dormant for a long time. Generally, this kind of bad traffic results in the greatest financial losses.

Another classification was proposed by Soto [16] where the unwanted traffic is categorized according to its primary or secondary sources. A primary source corresponds to the initiator of a communication such as when using a request like TCP SYN (TCP request with SYN flag set used to open a new connection), UDP, and ICMP Echo Request packets. Primary sources hence may include P2P services, spam email, viruses and worm propagation, intrusions, and massive attacks. Secondary sources correspond to traffic responses like TCP SYN/ACKs, TCP RST/ACKs, and ICMP. This class of unwanted traffic includes all traffic originated by backscatter and benign traffic.

In spite of important, the classifications of unwanted traffic seen so far do not consider one aspect: legitimate traffic. As previously mentioned, the discussion about what actually is unwanted traffic or not depends of where this occurs and the business model used. Considering this assertive, any traffic, even legitimate, that infringes one's business model is not welcome and labeled as unwanted. Based on this fact, this chapter presents another classification that also includes the legitimate unwanted type of traffic.

Unwanted traffic is therefore split into the following four categories:

- **Malicious codes** represent the kind of unwanted traffic employed purposely to damage initially hosts and consequently networks, without the user consent. Normally, these malicious codes steal data, allow unauthorized access to resources, exploit systems, and utilize the compromised hosts and networks to proliferate more unwanted traffic. This category is composed by viruses, worms, trojan horses (and variants such as remote access trojans and “gimme¹⁴”), and spywares. Among all of them, worms and trojan horses are considered the most “contagious” because of their capability of including unhealthy hosts in botnets (networks of invaded and controlled hosts), generate a massive spam, execute host service scans, and probe the IP address space. Malicious codes often exploit operating system and networking software design pitfalls and bugs. Like an epidemic, they

¹⁴ Actually, trojan horses are known as gimme, a slang for “give me”, in reference to spam messages that promise some gain or hot content.

propagate through networks at considerable speeds. The complexity of today's system software has opened the way for the increasing presence of this class of damaging traffic. To combat their spread, a number of fast response structures have been put into place to identify and provide network administrators with information on the recovery patches needed to cure their software. These sites should be consulted continuously to help administrators keep their system sane and stay a step ahead of attackers.

- **Unsolicited messages** represent a class of unwanted traffic used to send undesired and unsolicited bulk electronic messages. Email spam takes the biggest share. This is currently seen as the most cost-effective online advertising method available. This is very similar to the postal junk mail people are used to receive at home. A number of companies engage into aggressive email marketing campaign. A mail shot (or direct mail) is a form of advertising and apparently among the most successful ways of selling a product or service. Although they claim raising awareness among selected potential customers about their products, for most people it is nothing but junk mail. This is even worse in the case of the Internet as email addresses are very easy to collect from the Web using specialized crawlers and the cost of blind direct e-mail is minimal in addition to annoying the receivers and wasting network resources. Spam's bad reputation is due to the fact that email messages are often related to dozens of unwanted and illicit activities. Regarding illegitimate activities, spam scan type can be used for phishing techniques, social engineering attempts, "letter-bombs", denial of service attacks, mail storms, server overloads, and so on. With regard to unwanted activities, hoaxes, chain letters, and publicity are typical examples of how spam can be very unproductive. Another type of unsolicited messages is pop-up spam. Not discussed thus far, pop-up spam is a typical nuisance application since it exhibits windows messages like "error occurred" and "machine compromised". According to Krishnamurthy [11], this type of unsolicited message started at least 4 years ago and hundreds of millions of these messages are sent by hour. Spam variations include SPIT (Spam via Internet Telephony) and SPIM (Spam via Instant Messages). Spam creativity does not stop growing and there could certainly be more spam forms in the future using bulky objects such as video information.
- **Internet vulnerabilities** represent the kind of unwanted traffic generated or explored due to the design and building of the proper Internet or protocols that compose it. Denial of service attacks (DoS and DDoS) and its variants, attacks to the Internet infrastructure involving BGP (Border Gateway Protocol) and DNS (Domain Name Service), backscatter, low rate attacks, misconfigurations, and benign failures (outages and flash crowds) are examples of Internet vulnerabilities regarding unwanted traffic. The section 2.4 gives more information on Internet vulnerabilities. Architectural Internet services such as DNS and BGP were built with little concern with security and based on cooperation. In today's scenario even Internet state terrorism is not a fear-fetched possibility.
- **Recreational applications** represent the natural reason for the traffic growth seen on the Internet. The natural convergence among data (especially multimedia) associated with the user demands lead to an explosion of recreational traffic. Examples of such traffic include Internet radio, MP3 downloads, instant messages, interactive online games, and

specially streaming abundant multimedia traffic resultant from new applications such as Skype, MSN, Joost, Justin.TV, eMule, and Bit torrents. At first sight, the relationship between recreational applications and unwanted traffic is not perceived, but another deeper look changes that. Although typically this type of traffic is not related with the generation of gains for hackers, it is easily associated with the network resource waste like bandwidth and storage space, and the spent time of end-users in unproductive activities. For instance, online games, IPTV, and radio via Internet are not related with malicious activities, but are responsible for many bandwidth problems, especially at the edge and access networks. A recent Ipoque Internet study [23] reveals a considerable growth on web traffic. This fact is attributed to the popularity of file hosting, social networking sites and the growing media richness of Web pages. According to this study, file hosting has increased to up to 45% of all web traffic. Social networks¹⁵ are considered not productive traffic, chiefly in work environments, and can be used to spread malwares among users (SAMY¹⁶ virus, for example). The exception of the rule is P2P applications. Confessedly a large source of unwanted traffic, they carry malware codes, are responsible for high bandwidth-consumption, and are the main encourager for piracy because it breaches the copyright protection.

2.3. Unwanted Traffic: Who is guilty?

In today's Internet architecture, the presence of both malicious and unintentional unwanted traffic by some may be seen as a design weakness while others think that such phenomenon reflects human activities and behavior. There is truth in both statements. This section discusses the possible causes of unwanted Internet traffic about two different angles: Internet design principles and underground economy.

2.3.1. Internet Design Principles

At least 40 years ago, a set of desirable features was established to define the goals of an experimental network known as the ARPAnet, which major mission was to develop a robust military computer network using packet switching, a recent technological breakthrough invention of that time, capable to employ different link technologies (leased phone lines, satellite, radio, etc.) inside the same communication infrastructure, i.e., with internetworking support [25]. This network arises the Internet and these features were known as the Internet Design Principles and recorded in some important and fundamental papers and formal documents [25][26][27].

Although many years have passed, these design principles have withstood time by remaining remarkably stable, successfully resisting to a great number of new and emerging requirements from the different user communities. However, it is easy today to establish a relationship between them (as well as their existing implementations) and unwanted traffic. Basically, there was a total lack of concern with issues such as privacy, control, and security. The requirements were centered on the robustness of the

¹⁵ Social networks represent the interaction between human beings through the formation of groups or relationships. MySpace, Facebook, and Orkut are great exponents of social networks. Despite being ranked as the virtual world, the Second Life can also be seen in this category.

¹⁶ Samy virus is a worm, also known as JS. Spacehero, especially develop to attack MySpace social-network site [24]

routing relay network and its fundamental services were based on the assumptions that network members are going to cooperate in forwarding to the best of their capabilities information.

The first example is *layering principle* [26]. Especially created to insert simplicity and modularity in the Internet architecture development, the idea is that each layer, arranged in a vertical stack, relies on the next lower layer to execute more primitive functions and provides services to the next higher layer. This principle is known as hourglass model, where the IP protocol works as a universal data packet delivery mechanism (“IP over everything, and everything over IP”). However, this apparent simplicity has caused some problems. According to this model, all complexity was delegated to the network edges (end-points) while the IP layer (the thin waist of the hourglass) remained or tried to remain as simple as possible. Since the responsibility to implement all additional and necessary functions was “delegated” to the end-points, it is easy to understand the great number of vulnerabilities in protocols, software, and applications founded today. This is the price that one pays for giving user end systems more control over the network.

Specifically when speaking about unwanted traffic, the same idea has been used in P2P and multimedia applications to bypass traffic shaping policies. The Hyper Text Transfer Protocol (HTTP) has been used as generic transport protocol for applications that have little or no relation to the Web and its actual intended usage (Figure 2.1). Almost all P2P applications do take a ride over this protocol. The explanation is simple. HTTP is normally released in almost all firewalls, ACL (Access Control List), and filters. In addition, it is versatile and simple. Thus, it is easier to use and forge the HTTP communication with new applications instead of trying to surpass these security systems. As a result, the Internet infrastructure is also used to disseminate unwanted traffic.

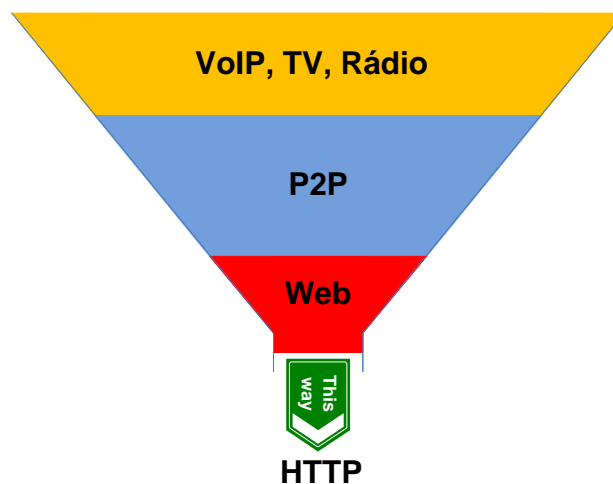


FIGURE 2.1: HTTP “hourglass model”

The *simplicity principle* [26][28] is the second example. The idea is that in order to be successful, the Internet needed to be as simple as possible. This philosophy is often referred to under the message: “Keep it Simple Stupid” or KISS for short. However, this choice also has influenced in unwanted traffic. In non-linear system theory there is the *amplification principle*. It states that there are nonlinearities that happens at large scales but which do not occur at small to medium scales [28]. In other words, small fluctuations in a dynamic system like the Internet may accumulate and produce major dangerously uncontrollable changes. The classic Internet example is the

BGP traffic. A recent case occurs involving the *youtube.com* and the Pakistan Telecom [29]. The Pakistan Telecom (AS 17557) accidentally started broadcasting internally an unauthorized announcement of the network prefix 208.65.153.0/24 as being that of *youtube.com*. However, one of Pakistan Telecom's upstream providers, PCCW Global (AS3491), forwarded this announcement to the rest of the Internet. It then took only few minutes to result in the hijacking of YouTube traffic on a global scale.

The last example is the *end-to-end argument* [26][28][30]. Basically, it suggests that communications protocol operations (functions) should occur at the end-points of a communications system or as close as possible to the resource being controlled. This reduces the subnet responsibility to packet relaying, or switching more accurately. The design of any new service does not suffer from any form of network processing and should be straight forward as the actual experience over recent years has indeed shown. However, there is ample evidence of the set of factors involving the end-to-end argument that directly corroborated with the unwanted traffic dissemination.

Firstly, this principle was planned to work in a small group of mutually trusting, trusted, collaborative, and technically knowledgeable and skilled users (end-points) attached to a transparent network [26]. Nowadays, this scenario is totally different and is practically impossible to guarantee that the original commandments of cooperation and willingness would be enforced. The statistics provided by different CSIRT (Computer Security Incident Response Team) teams ratify that the number of security events including attacks, inappropriate interactions (spam e-mail, for example), misconfigurations, and annoyances, increases day after day and practically always involves unsecure, uncompromised and badly used end-points. It is safe to assume that the Internet operates in an untrustworthy world when designing new services and protocols.

Secondly, instead of relying on dumb and limited terminals hooked onto reliable and often proprietary super and mini computers, today's users are computing consumers equipped with powerful personal devices and computers with embedded processors, portable user-interface devices, Web-enabled televisions and accessories, cell-phones, and so on. These users are not expected to understand the inner working of such powerful devices in order to use over networks. Consequently, configuration, protection, and control problems are trite and making these end-points easy targets to many types of security attacks, privacy invasion and other similar anomalies.

Lastly, the current business model of the Internet is not seen as being adequate to emergent applications. Actually, many ISPs view the massive use of streaming media and other types of new applications as a service to be offered only within some bounds (competitive differentiator) rather than a sort of capability to be provided, end-to-end, across multiple ISPs. The result is that while great investments have been made to keep the isolated networks, especially at the network core, reliable safe and trustworthy, the end-points are abandoned to their own faith. In addition, there is a trend of third-party involvement. Organizations, companies, and governments have demonstrated a growing interest in imposing in the communicating among end-points, to provide and enforce service accounting and taxation, law enforcement, and public safety. These interferences in the end-to-end argument can purposely insert unwanted and unsolicited communication between end-points.

To summarize, the IAB [10] enumerates some facts that prove how the purposely chosen natural Internet design principles have facilitated the rapid proliferation of unwanted Internet traffic:

- **Open nature:** The Internet is one of few operational platforms without control centers and surprisingly this feature is still contributing to its success. However, just like everything else, such open architecture comes at a cost. This design feature inflicts a series of technical limitations and problems as for example when identifying the whereabouts and identity of an attacker. The Internet architecture offers full and free communication between hosts and its main engine, the Internet Protocol (IP), does not provide any mechanism for auditing or taking a “tomography” of an attack. As a result, there is potentially no limit on what a host can do and there is also no record kept on the activities of a host by the network. This memoryless model has been purposely chosen for its simplicity and quick adaptability to network changes and traffic loads. In addition, the end-to-end argument [27] gave to the users a powerful mean to easily design and deploy new applications without the need for any network changes. Of course, harmful applications and unwanted ones have also taken advantage of this ride.
- **Versatility:** The same Internet infrastructure employed to access web pages, read e-mails, and chat is available for miscreants trying to get other types of benefits, often illegal ones, or simply cause harm to others. Protecting network resources and information has become a big business. A number of international and small players are offering their expertise and services to both enterprise level and home users. Think that any IT professional specialized in giving support and maintaining a network or a service (IRC, for example) is also potentially capable of eluding and skinning uninformed users and sometimes stealing and collecting privileged information or compromising the security of their hosts and systems.
- **Lack of meaningful deterrence:** There is no existing simple way to attribute responsibility when something goes wrong, be it unintentionally or maliciously. There is a limit to what existing solutions can achieve and continuous efforts are needed to seek and identify new threats and develop cures to these.

2.3.2. The root of all evil

In spite of the existence of many reasons to explain the real “boom” that unwanted traffic underwent in the last ten years, there is some consensus of the fact that there exists an infamous industry gaining lots of money with the generation and proliferation of unwanted traffic. This outlaw economy is responsible for stimulating and generating the most varied malicious activities such as stolen credit cards or bank accounts, malware and root kits design and spreading, phishing attacks, sale of logins and passwords, and so on, on the Internet.

This “black market” weighs heavily with the building and execution of malicious codes. Famous specialized security companies such as Kaspersky Lab [31] have perceived that current malwares and its variants exclusively aiming to obtain financial gains, and that “nonprofit” malwares are in extinction. Davies [12] asserts that this black market is strongly deep rooted into the Internet as a culture capable to dispose of billions of dollars around the world and that its eradication is almost impossible. The IAB workshop [10] considers this outlaw economy as “the root of all evil of the Internet”.

The foundation of this black market lays partially in the use of IRC servers. These are widely used to manage and execute illicit activities such as theft of bank accounts and credit cards numbers, login and passwords, and the proliferation of malicious code. In addition, IRC servers can be seen as big virtual multi-floor mall where stolen belongings and properties as well as private codes and tools are sold. For instance, in the first floor, equipments and tools (bots and botnets) for beginners are sold freely. On the second floor can be found more elaborated tools and hosts and routers access passwords. The third floor corresponds to retail sales. Bank accounts, credit cards numbers, and personal logins and passwords for ISPs and Internet services are negotiated directly or sometimes via auction sale. The last floor sold access to respectable servers of big companies and governments businesses.

Part of the profits is reinvested to diversify the illegal activities including the recruitment of professional writers to compose more elaborated spam messages, specialized programmers to develop new and robust malware codes (virus, worms, and spyware), Web experts (programmers and designers) to create more sophisticated Web sites for phishing activities. To complicate this ugly and dark scenario, the people behind this black market take advantage of the lack of punishment for most malicious activities and the lack for adequate legislation on the Internet. For instance, a DDoS attack usually makes use of a large number of hosts previously “corrupted”, spread across different backbones possibly in different countries. Since there is a large number of involved elements (hosts, access networks, backbones, routers, victims), it is not clear who should take the responsibility for the problem. Furthermore, the absence of a legal system in most countries, including Brazil, which criminalize some kinds of user conducts also, has contributed to the increase of malicious, non-productive, and unwanted traffic on the Internet. Even in countries where there is some jurisdiction over Internet crimes in place such as United States and England, the laws seek to penalize the violators only once the crimes have occurred and commonly much time after.

2.4. Recent Examples of Unwanted Traffic

In order to improve the understanding of the subject, this section splits the most notorious types of unwanted traffic in five subsections and presents for each one of them examples of attacks and related applications that are capable to generate unsolicited and unproductive traffic.

2.4.1. Internet infrastructure attacks

2.4.1.1. DNS

The Domain Name System (DNS) [32][33] is a hierarchical and distributed database that provides an essential service for the applications and Internet services: the translation of domain names to IP addresses. Due to its importance in the Internet infrastructure, any failure has potential to affect a large number of users and domains. This is behind the constant DDoS attacks made against DNS root servers. The most recent one was registered in February 2007 [34], when a significant Distributed Denial of Service (DDoS) attack, from the Asia-Pacific region, affected 6 of the 13 Internet DNS root servers that form the foundation of the Internet name service. Similar attacks have been occurring since 2002 [35][36][37][38].

Currently, there has been a considerable advance in the fight against DNS denial of service attacks. It has almost been mitigated due to efforts in developing and

implementing new solutions such as the anycast¹⁷ technology, for example. On the other hand, more specific and located DNS attacks gained more widespread. Among them, DNS spoofing attacks has been highlighted by many studies. The basic idea behind such attack is trying to corrupt the DNS information base, changing domain name data or adding new unreal domain addresses, aiming to redirect legitimate connections to fake servers (false addresses and websites) on the domains controlled by attackers. Actually, this type of attack is called pharming and its main target includes web sites for financial institutions.

There are two main techniques to perform DNS spoofing. In the first, known as DNS ID spoofing, the attackers run sniffers to intercept (Man in the Middle) DNS requests and get their request ID number. Next, a fake reply is sent with the correct ID number, but with the IP address of the corrupted DNS server. Recently, Dan Kaminsky [39] reported and publicly demonstrated a previously known BIND vulnerability implementation that permits attackers to “guess” the DNS ID and consequently to personify the DNS domain. DNS incidents occurred at the U.S. National Security Agency in May 2008 [40] and at the China Netcom (CNC) in August 2008 [41]. These were attributed to this spoofing vulnerability, but nothing was proved so far. The second technique is referred to as DNS cache poisoning. It is achieved through the invasion of a DNS server and replacing its DNS original information and data by other completely fake or modified data. The DNS zone transfer process is also used to spread the cache poisoning.

Another recent type of DNS attack is called Fast-Flux Domains. Typically, Fast-Flux Domains [42] presents quick changes of Resource Record (RR) data and therefore has a low TTL (Time to Live). Commonly, Fast-Flux Domains are composed by hundreds or even thousands of compromised hosts and are used to generate a Fast-Flux Service Network (FFSN). The attackers employ the services from such networks to send large volumes of spam via malware codes such as the WarezoV/Stration [43] and Storm [22] (and its variation including Nuwar/Zhelatin/Peacomm/Peed), to steal (phishing scheme) logins and passwords (in social networks like Myspace, for example). It is rather difficult to identify, track, and neutralize domains and hosts utilized for this type of illegal purpose.

Lastly, typo-squatter domains [44][45] and DNS rebind [46] are also used to spread unwanted traffic. Typo-squatting, also called URL hijacking, is a phishing technique where the attacker takes advantage of URL mistakes or typo errors. Attackers record and host unallocated domains with similar spelling to great access Web sites, changing only one letter or a sub-domain (from .com to .net or .org, for example). This type of action is known as domain parking and normally is used to spread virus, worms, adware and spyware. Recently, the work of Zdrnja *et al.* [47] identifies that many typo squatting domain are hosted in a same IP address to divulge advertisements and publicity.

DNS rebind tries to elude web browser to execute arbitrary malicious scripts on other machines of the same network [46]. Basically, this attack makes use of the DNS name resolution to forward request for a corrupted DNS server, configured to response with a TTL very low. The first response contains the server IP address that is hosting malicious code. Subsequent responses contain spoofing IP addresses to the attack target.

¹⁷IP Anycast is a load balancing and routing technology that enables multiple hosts to provide a service or function to a single IP address normally assigned to one host on the Internet. Servers that use IP Anycast share a single IP address, and user requests are routed to the nearest server on the network.

2.4.1.2. BGP

The Internet routing is based on a distributed system composed by several routers, aggregated in management domains known as Autonomous Systems (AS). These were introduced in the year of 1991 exactly when the Internet was moving from being a U.S government run project to embrace a new backbone that also interconnects to private and corporate networks. The main new change needed was to do with routing among these different networks. Hence, the Internet was split into politically and economically driven networks and there was a need to review how routing should be made between these new players. This leads to the birth of the Border Gateway Protocol (BGP) [48] as a solution for constantly advertising new reachability information among such ASs. These would then use the number of hops and other policies to decide on what external routes to use. Thus, Internet routing may be divided according to two distinct scopes: intra or inter domains. Attacks and anomalies concerned with intra-domain routing, though also relevant, do not produce chaotic effects, since the number of elements involved is often normally small. The management of intra-domain routing attacks is relatively easier to contain. On the other hand, routing attacks between different domains (inter-domains) are more worrisome because as they are likely to quickly affect all Internet traffic and clog expensive transatlantic links. Further, losing information or connectivity within one's domain is hardly the same as having one's traffic being wrongly forwarded or one's domain being unreachable.

Considered the main target of attacks and unwanted traffic, BGP [48], an inter-domain routing protocol was designed and implemented in the 80s and is seen as an Internet “*de facto*” standard today. However, according to the domain view at that time and similarly to what was deployed and operational on the Internet, security aspects were “minimized” in the BGP protocol design. Consequently, it is considered one of the five most vulnerable points of the Internet [49].

According to Kuhn *et al.* [50], BGP can suffer from many anomalies such as denial-of-service attacks, starvation, blackholing, delay, route looping, network partition, high churn, route instability, and router resource exhaustion. This is a part of the long list of BGP's vulnerabilities. Others include peers spoofing via TCP resets and ICMP, session hijacking, route flapping, routing disaggregation, malicious route injection, and so on. The consequences can be summarized in four results according to Nordström and Dovrolis [51]:

- **Blackholing**, where a prefix remains unreachable from a large portion of the Internet;
- **Traffic redirection**, where the traffic addressed to a specific domain is forced to take a different path towards a spoofed destination;
- **Subversion**, a special case of redirection, where the attacker forces the traffic to pass through some links in order to listen, spy or modify the data;
- **Instability**, resulting from successive advertisements (potentially with different attributes) and withdrawals for the same network.

In practice, the most notorious incidents involving BGP were caused by misconfiguration. What happened at AS7007 [52], AS3561 and AS15412 [53], and more recently to AS17557 [29] (Pakistan Telecom vs. YouTube) are examples. In addition, Anton Kapela and Alex Pilosov deployed a technique that simply exploits the natural way BGP works [54][55]. They make use of the naive trust BGP router put into each other when finding what they consider as best path and assert this with their

trusting neighbors. The technique uses IP hijacking to intercept data and advertise a range of IP addresses. The announcement would take just minutes to propagate worldwide, often before data heading to those addresses would even begin to arrive.

Unlike old IP hijack attacks where outages were created, this technique intercepts data silently to the actual destination, so here no outage occurs. This method is called AS Path Prepending that causes a select number of BGP routers to reject their deceptive advertisement. They then use these ASes to forward the stolen data initially destined to its rightful recipients.

2.4.2. SPAM

Spam (often defined as unsolicited commercial email or unsolicited bulk email) has grown dramatically in volume and in malefic results. Formerly, spam was synonymous to chain letters and its effect was time and resource wasting. Now, spam messages affect business productivity, inflate network traffic, and wastes storage space. This may sometimes lead to ISPs ignoring or the wrongful removal by users of useful electronic mail. Those who would find a definitive solution for spam email would make huge amounts of profit as almost everyone is affected by spam.

Spam messages can be classified according to their content. Hoaxes try to impress through fake histories and similar artifacts to guarantee their propagation (using urban legends, for example). Scam messages sometime lure recipients into offering them financial opportunities (such as huge lottery winnings and lucky winning draws). Phishing, typically uses commercial messages, and tries to obtain personal information (bank account details, credit card numbers and their passwords, for example) to be used in future frauds or shopping over Internet sites. Virus/Malware permits the installation of virus, worms, and trojan horses also to allow different types of fraud attempts or denial-of-service attacks. Pharmaceutical products, education, and adult content are also typical spam content.

Figure 2.2 illustrates a fairy spam offering an unusual service.

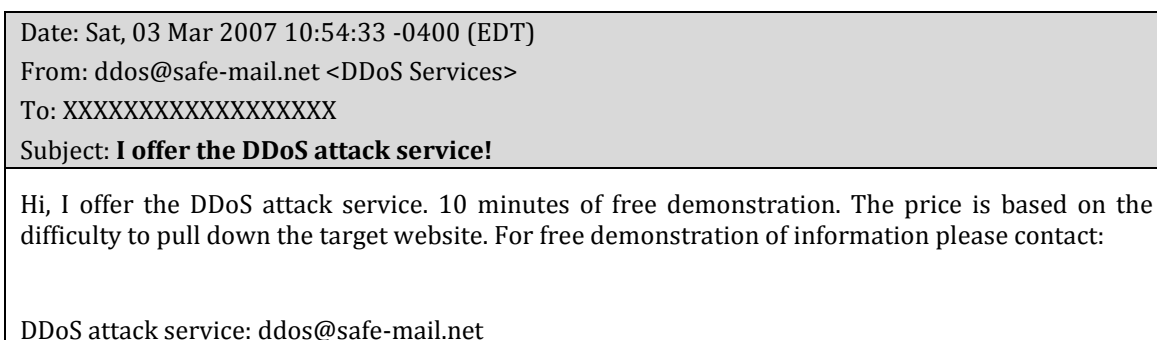


FIGURE 2.2: Spam email offering a DDoS attack service.

Despite that spam is hardly a novelty; it seems that current solutions have miserably failed in mitigating and removing this practice. Current statistics show that 2006, the Messaging Anti-Abuse Working Group (MAAWG) estimated that 80% of all email, based on an evaluation of approximately 390 million mailboxes, was spam [56] while the European Network and Information Security Agency (ENISA) asserted that almost two thirds of all emails that European providers received were nothing but useless spam [57]. To add to this wave of concern and complicate further this scenario, the Anti-Phishing Working Group (APWG), also in 2006, affirmed that at the time there were more than one hundred hijacked brands, several hundred unique password stealing

malicious code applications, more than one thousand passwords stealing malicious code URLs and up to ten thousand new phishing sites born every month [58].

2.4.2.1. SPIT and SPIM

Despite spam being the most famous type of unsolicited messaging, there are other types of nuisance unknown until recently and that began to cause considerable damage, namely, SPIT and SPIM.

Spam via Internet Telephony (SPIT) is characterized as unsolicited messages sent to VoIP users quickly and cheaply. Even spam traffic is also benefiting from recent cheap VoIP services. Among the first VoIP spam incidents are some that took place recently in Japan where VoIP usage is more common than elsewhere. In one of these incidents, spam announcements of an adult site were sent to unsuspecting SoftbankBB users, a major Japanese VoIP service provider. In a second effort, illegitimate requests for personal information were made [59]. Recently, similar incidents were reported in Australia and Columbia University [60]. Some Australian users reported that their mobile phones rang once and when they called back, they found themselves paying to listen to advertising material.

Unlike traditional spam email messages, which average only 10–20 kilobytes in file size, unwanted VoIP voicemails can require up to few megabytes of storage [61]. Although SPIT not having really taken root worldwide yet, an increasing number of incidents have already been registered. Recall that the level of spam was as little as 17% only in 2002. SPIT can be many times as annoying as spam. VoIP calls may be programmed to disturb the peace of people at home or while sleeping at night. Both the lawful intercept and tracing of VoIP calls is not trivial knowing Internet topology and privacy design concepts. Further, the presence of SPIT may lead to a huge drop in the quality of VoIP calls and consequently huge possible revenue losses. There is a justifiable fear of clogging phone lines with unlawful advertising and spam messages.

Spam via Instant Messages (SPIM) represents the delivery of unsolicited messages sent by instant messaging applications. Generally, this type of spam is based on the creation of fake profiles in instant message systems to send unsolicited messages, which could include commercial scam-ware, viruses, and links to paid links for the purpose of click fraud. SPIM is usually sent in the form of request messages that cause content to automatically appear on the user's display. The typical request messages in SIP (Session Initiation Protocol) [62] are as follows:

- SIP MESSAGE request (most common)
- INVITE request with large Subject headers (since the Subject is sometimes rendered to the user)
- INVITE request with text or HTML bodies

Figure 2.3 shows examples with SIP INVITE and MESSAGE.

```
INVITE sip:Bob1@192.168.10.10:5060 SIP/2.0
Via: SIP/2.0/UDP 10.10.10.10:5060;branch=z9hG4bK00002000005
From: Spammer <sip:spammer1@10.10.10.10:5060>;tag=2345
To: Bob <sip:Bob1@192.168.10.10>
Call-Id: 9252226543-0001
CSeq: 1 INVITE
Subject: Hi there, buy a cool stuff in our website www.spam-example.com
Contact: <sip:spammer1@10.10.10.10>
```

Expires: 1200 Max-Forwards: 70 Content-Type: application/sdp Content-Length: 143
MESSAGE sip:Bob1@192.168.10.10:5060 SIP/2.0 Via: SIP/2.0/UDP 10.10.10.10:5060;branch=z9hG4bK00002000005 From: Spammer <sip:spammer1@10.10.10.10:5060>;tag=2345 To: Bob <sip:Bob1@192.168.10.10> Call-Id: 9252226543-0001 CSeq: 1 MESSAGE Max-Forwards: 70 Content-Type: test/plain Content-Length: 25

FIGURE 2.3: SPIM example.

2.4.3. Malicious code

Malicious code represents the unwanted traffic used to cause damage to computers, systems, and networks. Typically, it steals data, allows unauthorized access, exploits systems, and the use of compromised computers and networks to proliferate further unwanted traffic.

The main examples of malicious code are:

- **Virus:** a program that can copy itself and infect a computer without permission.
- **Worm:** a self-propagating piece of malicious software that spreads across a network.
- **Trojan:** a destructive program that masquerades as a benign application.
- **Bot:** a program used for the co-ordination and operation of an automated attack on networked computers.
- **Rootkit:** a set of programs that work to subvert control of an operating system from its legitimate operators by making changes to the underlying operating system itself.
- **Spyware:** a program installed surreptitiously to intercept or take partial control over the user's interaction with the computer.
- **Backdoor:** a method of bypassing normal authentication obtaining covert access to a computer, while attempting to remain undetected.
- **Downloader:** a program that downloads and installs malicious software.
- **Adware:** a package that automatically displays or downloads advertising material to a computer.
- **Ransomware:** a type of malicious code that encrypts the data belonging to an individual on a computer, demanding a ransom for its restoration.

2.4.3.1. Botnets

Among all malicious code related with unwanted traffic, bots are the more alarming ones due to their capacity to cause disastrous effects on the worldwide network infrastructures. According to some estimates, there are nowadays between 500 and 2550 different botnet Command & Control (C&C) servers running every day [63][64]. This is

actually a growing business as some are offered for rent to attackers interested in disturbing company sites and even official overseas government institutions. However, due to the development of countermeasures, botnets have been changing to adapt to new technologies and contexts. For example, instead of using the IRC protocol, HTTP became the preferred protocol to create these new plagues. Another example is the Storm botnet. It uses a similar protocol to P2P (and UDP over port 4000) to establish communication with other peers.

Unlike other malwares, botnets can be used in the most diverse malicious activities like DDoS, spam, and frauds. For example, assuming that a botnet has an average of 20.000 hosts (bots or zombies) and that a single DDoS attack session can consume 40Kb/s of upload bandwidth from each bot, the consequences of this attack can be enormous if one does the mathematics. Online frauds can also be performed by botnets. Once installed in a host, a bot sends personal information (login credentials, bank accounts, intranet applications, webmail, online services, social web pages, and so on) and exploitable information (installed programs' serial numbers and online gaming credentials, for example) to its C&C server that should use them in different unlawful activities.

Also taking advantage of thousands of controlled bots, botnets are especially used to send spam. A spam bot can send up to three spam e-mails per second (259.200 e-mails per day). In addition to making it difficult to track down the spam sources, spam e-mail can contain scam, illegal pharmacy sites or the fraud known as 'pump-and-dump' (or 'stock spam'), involving the use of false or misleading statements to hype stocks, which are 'dumped' on the public at inflated prices.

Lastly, botnets can distribute malicious code to infect new bots and permit the use of malicious software (Internet banners or advertisements, for example).

2.4.4. Social networks

Social networks¹⁸ (Bebo¹⁹, Facebook, MySpace, Orkut, and Hi5²⁰ are famous examples) can be considered the new Internet phenomenon. The growing number of users of these services at a drastic rate in the last few years is proof of that. For example, in June 2010, Facebook announced had 500 million users around the world achieves [65].

In general, such large number of users is attracted by a set of functionalities focused in the social relationships such as posting personal data into profiles and the creation of a "circle of friends" sharing common interests and life styles. Furthermore, social networks are used as a forum for collaboration, education, experience-sharing, and trusted information exchange.

Despite their apparent harmless face, social networks may present innumerable risks to their users. In [66], the authors describe many of these. Among the most relevant to unwanted traffic there are:

- **Spam:** due to the exponential growth of social networks, spammers begin to invest in these networks to spread unsolicited messages to their users. In general, they use specific spamming software such as FriendBot [67] to

¹⁸Social networks are often called as online social network sites or social networking sites.

¹⁹ <http://www.bebo.com>

²⁰ <http://hi5.com>

automatically send friend invitations and note/comment posting. Typically, these messages can include links to product sites, adult content or phishing.

- **Cross site scripting, viruses and worms:** the use of HTML to create profiles and/or post messages leaves the door open for new attack possibilities. Cross-site scripting²¹ (XSS) attacks and viruses like SAMY recently infected one million MySpace user profiles in less than 20 hours [69].
- **Spear phishing:** since there are millions of user profiles (and “circles of friends”) completely and easily available in social networks, phishers have developed highly targeted phishing attacks known as spear phishing. For instance, the worm JS/QuickSpace.A [70] infected MySpace profile pages with links to a phishing site to steal their profiles.

To sum up, the benefits offered by social networks is visible and fundamental for the development of society, but if it fell into the wrong hands (spammers, attackers, and phishers) these powerful tools can produce catastrophic results and huge amounts of unwanted traffic. In addition, social networks affect the productivity, since the employers involved in social networks waste part of their time checking these circles to keep updated.

2.4.5. Recreational traffic

Unwanted traffic generated by recreational applications is a consequence of real Internet progress. It represents the natural convergence between the different types of data, especially multimedia, associated with the growing number of users and their interest and focus on the news. Examples of recreation traffic include radio and television via Internet, P2P file sharing, video sharing (YouTube, for example), instant messages, and online games. Social networks are also responsible for generating recreational traffic, but they will not be discussed again in this section.

Regarding to unwanted traffic, recreational traffic is responsible for a large chunk of bandwidth consumption, especially at the edge network. In other words, this type of traffic steals bandwidth that should otherwise be used for business and what some may refer to as “useful” applications. Recreational applications are very aggressive and may quickly start consuming large amounts of WAN and Internet bandwidth. To download quickly, a stream of “live” video or swap files efficiently, recreational applications may initiate a large number of simultaneous connections and start suddenly consuming large amounts of bandwidth resources during sustained periods of time [71].

For example, the YouTube, currently considered as the most famous video sharing website, needs to convert upload videos into .FLV (Adobe Flash Video) format after uploading these. According to Michael Dell of Dell Inc. [71], YouTube traffic in 2007 consumed as much bandwidth as the entire Internet utilized just seven years ago. In spite not using a P2P system, YouTube video files are located all around the world. In addition, they are not streamed. Contrariwise, they are downloaded and buffered.

Another example is represented by P2P file sharing applications. Basically, these applications portray some natural features that confirm their tight relationship with

²¹Cross-Site Scripting (XSS) is a type of computer security vulnerability typically found in web applications which allow code injection by malicious web users into the web pages viewed by other users [68].

unwanted traffic. File transfer, may consume as much bandwidth as there is available. Bidirectional traffic and aggressive behavior (sharing different contents in the same bandwidth) are some of the features that clearly explain the “power” of P2P file sharing. In addition, they are confessedly source of main malicious code spreading mechanisms. They have been responsible for spreading viruses, Trojan horses, and bots. Moreover, there is a major problem with the use of current P2P file sharing: that of copyrighted content (music, video, books, and so on.). The main reasons for this concern stem from the natural distributed feature of P2P networks and the lack of laws to govern and enforce policies for their worldwide content exchange. Consequently the volume of unlawfully used copyrighted material across P2P networks continuous to increase on a yearly basis.

2.4.5.1. Encapsulated and obfuscated traffic

Newer versions of P2P protocols can flexibly use any port number, even port 80. This technique is called port hopping. The cat and mouse race does not stop here with regard to P2P file exchange. A surprising recent Internet development caught by surprise the community. Many P2P applications are now giving their users the optional luxury of allowing them to intentionally hide or camouflage their traffic.

In order to avoid recent payload string matching and signature based detection methods, P2P applications have been working on the fast track to use encryption and SSL. This way such encrypted traffic would be missed (unrecognizable) as P2P content. This counter-technique is called protocol obfuscation²² [72]. It is employed to surpass traffic shaping limitations that found their application in many Internet providers. It permits the hiding of the protocol structure (data and control messages through their encryption). For example, in the case of the eMule P2P application, the use of this technique changes all communication data to appear just like a random data, hence complicating its identification and consequent mitigation. Recent examples of protocol obfuscation in P2P applications have also been seen with both the BitTorrent and Skype P2P applications.

2.5. Chapter Summary

In this chapter, the universe of unsolicited, non-productive, irrelevant, and illegitimate traffic that crosses the Internet on daily was introduced. First, a more detailed view of the unwanted Internet traffic was presented through some definitions. Due to the presence of a number of sometimes limited definitions that were adopted in both academia and the industry, a generic one embracing varying relevant characteristics for unwanted traffic was given. Moreover, context problems are also presented. After, a formal classification that was proposed during the IAB workshop on unwanted traffic and others based on its traffic sources were discussed. The failures of such classifications were shown and a new and more general classification was made. It includes most known types of unwanted traffic as well as considers even legitimate traffic that can be considered unwanted in some contexts.

Next, the possible causes of unwanted Internet traffic were discussed. First, some Internet design principles and how they have been exploited to create and spread attacks and traffic anomalies have been discussed. Three of the most important Internet

²² *Protocol obfuscation* also is known as *protocol encryption*, *message stream encryption*, *protocol header encryption*.

principles: layering, end-to-end argument, and simplicity have also been described. Examples showing their limitations and the way they have been explored have been presented as a proof. To finish, some disturbing reports on the existence of an underground economy behind unwanted traffic were put forward. Many services taken for granted by Internet users today, such as IRC, have their servers being utilized as marketing platforms for where everything related to fraud, stolen information, and security invasion can be traded.

Towards the end of this chapter, the main types of unwanted traffic were presented. First, many Internet infrastructure attacks, including denial of services (DoS and DDoS), DNS, and BGP anomalies were described. In order to provide the reader with a more complete view, recent incidents, events, anomalies, and attacks witnessed across the Internet community were presented. Secondly, spam, unsolicited messages, and its variants SPIM and SPIT were shown. The alarm has been raised for taking steps before SPIT simply dominates VoIP traffic, as was the case with spam dominating email traffic. Thirdly, malicious codes were presented and a special section was dedicated for botnets. Some of its relevant features and examples were showed. Fourthly, the unwanted traffic generated in association with social networks was discussed. The main issues and risks this new phenomenon faces have also been presented. Lastly, recreational traffic was discussed including encrypted and obfuscated traffic.

Chapter 3

Approaches against Unwanted Traffic

In the search for the Aladdin's lamp to deal with unwanted traffic, a considerable number of solutions have been employed to identify and mitigate its effects on the Internet traffic.

This chapter reviews some of these solutions. First, traditional and well-known tools, including firewall, IDS, anti-something software, and honeypots, are presented and its pros and cons are discussed. After, promising solutions are explained. These solutions are not really utilized to stop unwanted traffic, but can help substantially on the process of identification. Next, collaborative solutions, those capable to mix different approaches and tools, are presented. In this section, the interest is on the necessary requirements and features to project and develop an effective collaborative solution. Finally, recent traffic analysis approaches and strategies are reviewed to show its applicability on unwanted traffic detection.

It is not the aim of this chapter to claim that all of the existing strategies are reviewed here but it should give a good idea on current work and argue for the need to continue such efforts as a solution remains a further undertaking.

3.1. Traditional Solutions

3.1.1. Filtering

Undoubtedly, filtering mechanisms are among the most widely deployed security solutions in the world and can be considered the first line of defense.

Traditionally, filtering mechanisms are represented by firewall, Access Control List (ACL), proxies, and application-level gateways. BGP null routing, a DDoS mitigation technique very popular for ISPs, consists in the changing of every edge router to configure null-route and consequently to stop a victim host attack [73], also can be considered a traditional filtering mechanism. The main function of these mechanisms is to approve or deny the traffic exchange between networks. Basically, they employ rules that define what to do. This way, all ingress or egress network traffic match with rules and, as result an action is taken.

However, their effectiveness has not been sufficient against undesired traffic. Some aspects prove that. First, although practicable, they are inevitably imperfect because it mainly relies on "heuristics" and manual configuration to identify unwanted traffic. Consequently, they can harm both unwanted and legitimate packets. Second, they require application-specific support. Filtering mechanisms like proxies or application-level gateways are developed to evaluate the traffic (encrypted or not) of specific application. This way, for each new service or application a new specific solution needs be build.

3.1.2. Intrusion Detection System

Intrusion Detection Systems (IDS) are hardware and/or software designed to detect unwanted attempts at accessing systems or networks [19]. Basically, IDS solutions are composed by sensors that generate and send events and security alerts to management stations.

Traditionally, signature-based (or misuse-detection) and anomaly detection are the two most used approaches to build IDS. Signature-based strategies identify patterns matching network traffic or application data using an attack signature, often in the form of a known bitstring, from a previously compiled database. Known attacks are detected fairly quickly with a low false positive rate, while unknown ones often slip through. Nonetheless, anomaly detection is a more generic approach. It works based on building a behavior profile for what is considered as normal activity which is then matched to the actual traffic in order to find out anomalous events. Hence, this class of approaches is capable of adapting to new classes of anomalies as well as “zero day” attacks. This is seen as powerful advantage over other techniques.

Although mostly capable of discovering a wide range of malicious activities, by definition, IDS are passive, that is, only detect and record events. No action is taken. Moreover, they suffer from accuracy problems: false positives and false negatives. In addition, IDS are focused mainly only on internal security.

3.1.3. Anti-something software

The closest solution to end users can be referenced as the anti-something software. It represents all programs designed to detect and remove potential threats to systems and networks such as anti-virus, anti-spyware, anti-phishing, and anti-spam.

Antivirus are software that detect and remove computer virus. However, its usefulness depends on constant update, since daily new virus and/or variant of know virus are spread. Other issue regarding antivirus is what to use. There are a great number of available antivirus solutions, differentiated by features like detection method, offer functionalities, and price.

Anti-spyware software is used to fight against software and spy codes such as spyware, adware, and keylogger. Similar to antivirus, there is dozens of solutions split in commercial and free. However, as aggravating, some anti-spyware solutions are famous to spread spyware [19]. Anti-phishing software aims to block possible fraud attempts in web sites or e-mail. Typically, this type of solution is embedded on web browsers, email clients, and toolbars. In spite of to help the end users, there are some issues about the current versions. According to Wu *et al.* [74], the location of toolbars and information display does not favor end users, and there is not any suggestion about to do when a phishing attempt is detected.

Lastly, anti-spam solutions try to detect spam based on filtering unsolicited messages through the header fields or message content. Header fields filtering checks source address, name of the sender, and subject of a message to validate it or not. This type of anti-spam solution is simpler, but is more prone to setup errors, since it is necessary to define rules that whether or not to receive, or which addresses, senders and subjects are unwanted. Blacklists (blacklists) are examples of filtering header. On the other hand, filtering based on message content is the most used. Usually this technique performs searches for keywords in the content of messages. When configured correctly,

the content-based filtering is very efficient, but also can make mistakes. Furthermore, the content inspection does not check the origin of the message.

3.1.4. Honeypot

The term honeypot refers to a security tool whose main function is to collect information about attacks and attackers, i.e., a software or system that has real or virtual security failures intentionally implemented and with the purpose of being invaded and attacked so that the used invasion mechanisms can be observed and studied.

Honeypot have been used against unwanted traffic to characterize traffic and advise network managers and operators (beyond network devices such as firewall and IDS) about attacks and anomalies (bots and worms using IP address space not allocated or not allowed, for example). In addition, they offer traffic trends that can be used to take decisions about the network security. Currently, *honeyd* is most famous open source honeypot available [75]. However, it is important emphasize that honeypots do not provide any kind of prevention, but offer invaluable information for use to build a cure to the attack being observed.

3.2. Promising Solutions

Although traditional solutions are being employed against the unwanted traffic with a certain level of efficiency, currently, some newly developed solutions such advanced filtering and IP space investigation can be used to improve the effectiveness of traditional solutions.

3.2.1. Advanced Filtering

As previously mentioned, the typical filtering is not sufficient for dealing with the current unwanted traffic level. In order to address this problem, researches have proposed advanced filtering schemes and mechanisms. Currently, the IETF (Internet Engineering Task Force) best current practices on network ingress filtering BCP 38 (RFC 2827 [76]) and BCP 84 (RFC 3704 [77]) are pointed out as the effective solution to block DDoS attacks using spoofed source IP addresses.

BCP 38 is a filtering method that prohibits attackers from using forged source addresses which do not reside within a range of legitimately advertised prefixes [76]. In other words, if an ISP is aggregating routing announcements for multiple downstream networks, strict traffic filtering should be used to prohibit traffic which claims to have originated from outside of these aggregated announcements [10].

BCP 84 is focused for multihomed networks and presents other ingress filtering implementations such as Strict Reverse Path Forwarding (SRPF), Feasible Path Reverse Path Forwarding (Feasible RPF), Loose Reverse Path Forwarding (Loose RPF), and Loose Reverse Path Forwarding Ignoring Default Routes, which offer automatic and dynamic configuration of advanced ingress filters mechanism in core and transit networks. Nowadays, the joint of these practices with Remotely Triggered Black Hole (RTBH²³) filtering technique is being proposed by IETF draft [78] to drop unwanted traffic before it enters a protected network.

²³ RTBH filtering is a technique that uses routing protocol updates to manipulate route tables at the network edge or anywhere else in the network to specifically drop undesirable traffic before it enters the ISP network [79].

In [10], researches and specialists argue that a global deployment of BCP 38 and BCP 84 will permit effectively to block DDoS attacks using spoofed source IP addresses. However, the lack of incentive, infrastructure changes, and mainly the possible blocking of legitimate traffic due to accidental errors are some of reasons for a lack of a wider deployment.

3.2.2. IP address space investigation

The investigation of the traffic coming from unassigned, unused, and unannounced IP address spaces is another potential solution, since these addresses are used in unwanted activities such as scanning, DDoS attacks, and worm and virus infection.

Currently, Internet Motion Sensor (IMS) [80][81] and Network Telescopes [82][83] are the most important implementations. Internet Motion Sensor (IMS) is a globally distributed monitoring system which goal is to identify and track traffic native of or from those types of address spaces and non-routable networks. According to [10] the IMS monitors approximately 17 million prefixes, about 1.2% of the IPv4 address space spread around the world in ISPs, organizations, enterprises, and universities. Network Telescopes are composed by monitors keeping an eye on traffic routed to unused IP address space and observe Internet events like viruses and worms propagation. The idea behind a network telescope is to maintain active hosts to listen all traffic sent to these address spaces. This way, it permits to see exactly all events in their “brute state”, i.e., without any traffic interferences.

So far, IP address space investigation is not an unwanted traffic tool or solution. Its results have been used to gain knowledge about the actual propagation and effects caused by attacks and malware. Hereafter, it will be used to help in the fight against unwanted traffic.

3.3. Collaborative Solutions

The term collaboration is defined as “mutual engagement of participants in a coordinated effort to solve the problem together” [84]. Applied on network security area, collaboration is the process of detecting abnormal behaviors or events by a group of security solutions and devices that share information with each other. More specifically, collaboration permits employ different approaches, solutions, and tools to deal with the most vary types of anomalies and attacks.

The use of collaborative solutions has a more effective due to the arising of more and more elaborated and coordinated attacks and anomalies such as worm infections such as SQL-Slammer [85] and Storm [22][86] worms and distributed attacks as the DDoS attacks occurred from September 2007 to March 2008 [87].

Typically, collaborative solutions are composed by misuse and anomaly based Intrusion Detection Systems (IDS). The misuse normally checks for intrusions at packet level over single connection and are capable to detect known attacks fairly quickly with a low false positive rate. The latter, anomaly, is a more generic approach, which is based on building a behavior profile for what is considered as normal activity and later on matched to the actual traffic in order to find out anomalous events. The last one is also recongnized as Anomaly Detection Systems (ADS) and are capable of adapting to new classes of anomalies, as well as “zero day” attacks and are mostly observed at the network level involving multiple connections.

Based on these features and advantages, its integration in collaborative solutions can permit not only the detection of known attacks but also unknown anomalies. These approaches are known as Collaborative Anomaly and Intrusion Detection Systems (CAIDSs) or as CIDS (Collaborative IDS) and DIDS (Distributed IDS).

The goal of a CAIDS solution is to permit that distinct detection systems work jointly and cooperatively, allowing traffic anomalies identification quickly and accurately through the reduction of the numbers of alerts, the discard of false alerts, and a global view of the anomaly. Normally, CAIDSs are composed by detection units, formed by multiple detection sensors, where each sensor monitors its own sub-network or hosts separately and then generates low-level intrusion alerts; and at least one correlation unit to transform the low-level intrusion alerts into a high level intrusion report of confirmed anomalies.

In spite of its advantages, CAIDS solutions introduce new challenges or requirements for detection activity such as normalization, aggregation, and correlation of alerts, false alert reduction, prioritization, and prediction. The subsequent sections discuss each one of these issues and present some solutions for them.

3.3.1. Alert Normalization

Since collaboration involves different detection systems and typically alerts encoded in distinct and proprietary formats, the use of standard message formats and/or protocols to information exchanges (data and control) among them is a key aspect with important impact on the collaboration scheme.

Recent efforts resulted in three standards for information exchanges among detection systems. The first is the Intrusion Detection Message Exchange Format (IDMEF) [88], an XML based specification for an intrusion alert format, which defines data format and exchange procedures used to exchange information between detection systems and management centers, independently of the communication protocol. The second is the Intrusion Object Description and Exchange Format (IODEF) [89], which defines a data representation (format) and a framework for CSIRTs to exchange operational and statistical security incidents information among themselves. In addition, IODEF was designed to be heavily based on IDMEF and provides upward compatibility with it. The last of recent standards is the Intrusion Detection Exchange Protocol (IDXP) [90], an application-level protocol for exchanging data between detection systems and focused in to provide exchange of IDMEF messages, unstructured text, and binary data, beyond to support mutual-authentication, integrity, and confidentiality over a connection-oriented protocol.

The use of these standards provides a series of benefits, including representation of alerts in an unambiguous fashion, interoperability among different tools and systems, facility to aggregate alerts, and capability to establish correlations among them, improving the accuracy of detection process.

Currently, IDMEF language has been adopted as standard in great part of works such as [91][92]. However, according to Sadoddin and Ghorbani [93], IDMEF standard presents still some issues to be addressed for true inter-operability among detection systems. First, IDS normally use different nomenclatures in order to fill the classification fields of alerts. In other works, there is a lack of common convention to fill alert information. Second, there is no accepted taxonomy to describe attacks.

Typically, each detector have its own definition mapped locally, what limit or difficult the interoperability with other detectors.

3.3.2. Alert Aggregation

Regardless of the use or not of some kind of alert normalization, the activity of alert aggregation must be essential in any proposed CAIDS solution. According to Sadoddin and Ghorbani [93], the use of alert aggregation is justified by two reasons. First, similar alerts tend to have similar root causes or similar effects. Second, due to large number of alerts produced by low-level sensors for a single malicious activity, alert aggregation has proven to be highly effective in reducing alert volume.

Normally, alert aggregation is made matching the similarity between determined numbers of attributes (alert fields) except with little time difference. A good example is the work of Valdes and Skinner [94], which utilizes IP addresses (source and destination), port (source and destination), time, and attack class attributes to extract similarities among alerts. However, recent works have been extended this concept employing the use of cluster to group alerts that share the same root causes, due to the fact that this type of organization permits to easily detect causality or false positives on the analysis. The works of Julish [95] and Cuppens [96] are interesting examples. The former aggregates all alerts, which share the same root, causes what is intuitively the reason for which alerts occur. For this, hierarchy structures, called generalization hierarchy, are used to separate the attributes of alerts from the most general values to the most specific ones. This way, dissimilarities of two alerts can be measured comparing the longest path between values of that attribute in the corresponding structure. The latter employs a relational database to store alerts and evaluate them using a set of expert similarity rules to group them according to the occurrence in a same attack. Other works as Morin *et al.* [97] and Xie *et al.* [98] are also based on clustering techniques.

Other works consider that alert aggregation is a function of alert correlation. Zhou *et al.* [99] and Yusof et al [100] classify aggregation (similarity and clustering) activity as similarity based approaches for alert correlation due to the fact that these techniques are intrinsically related with design and implementation of alert correlation function.

3.3.3. Alert Correlation

Alert correlation has the function of to detect attacks and anomalies, in different stages, and produces a high-level description of the abnormality on the network. According to Sadoddin and Ghorbani [93], the goal is to find causal relationships between alerts in order to reconstruct attack scenarios from the individual alerts.

In spite of there is a discordance whether alert correlation include aggregation activity, this thesis shares the view presented in other works [99][100] and divide alert correlation into the following categories: similarity based, attack scenario based, rule based, and statistical based.

Similarity based techniques

Similarity techniques are typically based on the similarity between alert attributes. Basically, they compare an alert to all alerts that have similar attributes or features (e.g. source IP address, destination IP address, ports, time, attack class, and so on). Similar

alerts tend to have similar root causes or similar effects on network resources. For this purpose, techniques like similarity and clustering are employed.

Valdes and Skinner [94] have developed a probabilistic alert correlation approach for EMERALD [101] project. They implement three phases of correlation: *synthetic attack threads*, where the alerts are clustered if some similarity is found; *security incidents*, used to fuse the same attack reported by multiple detectors; *correlated attack reports*, which merge alerts representing different stages of a complex attack. In [102], Debar and Wespi proposed an aggregation and correlation algorithm for intrusion alerts. In this approach, three steps are necessary to perform alert aggregation and correlation: *alert processing*, where the alerts are translated to a data model (the authors used the first discussion about IDMEF as data model); *relationship correlation*, that extracts correlation between alerts; *relationship aggregation*, where the output of second step (alerts) are aggregated into seven different scenarios (situations) according to its attributes.

Regarding clustering approaches, the works of Julish [95] and Cuppens [96] are considered very representatives. Already the work of Zhu and Ghorbani [103] employs two neural networks approaches (Multilayer Perceptron and Support Vector Machine) to determine correlation between alerts and consequently establish causal relationships. For this, they introduce the idea of Alert Correlation Matrix (ACM) to store the average correlation between alert classes, which is computed adaptively based on statistical analysis of consecutive input alerts. The adaptation characteristic of this method makes it possible to start with initial (and maybe immature) correlation probability values and learns more from the environment as the operation proceeds, and help to extract high level attack strategies.

Attack scenario based techniques

Attack scenario techniques use the fact that attacks often require several actions or steps to take place in order to succeed [99]. The idea is that every attack scenario has corresponding steps required for the successfulness of the attack. This way, low-level alerts can be compared against attack scenarios before the alerts can be correlated.

Typically, approaches using this technique have used formal models defined by expert users for specifying attack scenarios or employing machine learning to create attack scenarios. In [104], Morin and Debar propose a multi-alarm misuse correlation component based on the chronicles formalism able to model attack scenarios. The chronicle formalism proposed by Dousson [105] is used to build correlation blocks and represents a set of patterns (attack scenarios), whether new alerts are received; they are compared with chronicles. Chronicles are update always a matching occurs or otherwise are constructed.

In addition, researches have proposed several formal correlation and definition languages to generate attack scenarios. Among the most known are LAMBDA [106], STATL [107], ASL [108], JIGSAW [109] and ADeLe [110]. In [111], Dain and R. Cunningham propose a scheme to fuse alerts into predefined attack scenarios. The idea is to use a fusion system to determine for that attack scenario the alert belongs. Thus, always that a new alert is received it is compared to determine for what attack scenario the alert must be a member. The scenarios are generated using two approaches, one based on heuristic and one based on data mining.

Rule-based correlation techniques

Since attacks and its variants should be generating a large number of scenarios, the use of rules (pre-condition and post-conditions) has been employed to address this problem, reducing the number of possible attacks scenarios. This approach is known as rule-based correlation techniques and some authors to classify it as subclass of attack scenario based technique.

The paper of Debar and Wespi [102], previously described and classified in similarity techniques, uses consequence rules to define attack scenarios. Consequence rules specify that one event (alert) should be followed by another type of event, allowing thus that alerts be correlated. In [112], the authors proposed an approach to map causal relationships between alerts using rules. They introduced the concept of *hyper alerts* to encode the pre-condition and post-condition of an alert in its prerequisite and consequence fields, respectively [93]. This way is possible to extract prerequisites and consequences of hyper alerts, generate graph, which is useful in determining the attacker's goal.

Statistical techniques

Despite effectiveness, similarity and attack scenario techniques are only focused to correlate known attacks and anomalies. In order to complete this deprivation, statistical techniques have been proposed to detect unknown attacks and anomalies.

Qui and Le [113] use Granger Causality Test (time series analysis method) to correlate alerts which emphasis on attack scenario analysis. The idea behind this approach is to use the causality analysis to correlate alerts and generates attack scenarios without any pre-defined knowledge. For this, it assumes that each multi-step attack will generate alert that have statistical similarities in their attributes, and this attack steps have causal relationship [100].

In other work, Qui [114] employs a Bayesian network to model the causality relationship between alerts, where the alerts are node and its causality relationships are edges. In this model, continues alerts are divides along equal time slots and the state of each node corresponding to an alert is a binary value representing the presence of the alert in the time slot [93]. The central idea of this work is to discover which alert types may cause an alert of type X and how the conditional probability of X is related to its causes (parents). Almgren *et al.* [115] also use a similar approach.

3.3.4. False Alert Reduction

The recent detection systems have faced a serious problem caused by the large number of alerts. However, the problem is not only limited to huge in quantity, but also in quality with the presence of a high rate of false alerts. Basically, this occurs because the detectors employ different approaches (algorithms, information bases, and rules) to find a determined type of attack or anomalies. This way, to reduce the false alerts aiming to identify real alerts becomes an essential requirement for the deployment of any CAIDS.

Several methodologies have been applied to solve the problem of false positives. Alharby and Imai [116] proposed to mine historical alerts aiming to discover how future alerts can be more efficiently handled. The proposal consists in, firstly, to characterize the "normal" flow of alerts and, lastly, an algorithm for detecting anomalies based on continuous and discontinuous sequential patterns. Viinikka and Debar [117] make use

of Exponentially Weighted Moving Average (EWMA) control chart for extracting trends and highlighting abnormalities on the alerts flows.

In [118], Manganaris *et al.* use Frequent Episode Rules (FER) [119][120] to create profiles from non-intrusion periods. The use of FER allows discovering real alerts in the future, since sequence of alerts matching a frequent episode rule is considered as false positives. Clifton and Gengo [121] also use the frequent episodes to discover usual sequences of alarms through time and characterize them as false alarms.

Yu and Frincke [122] propose the use of Weighted Dempster-Shafer, an extension from the basic Dempster-Shafer theory, to combine beliefs in certain hypotheses (e.g. alerts reported by a signature-based sensor and an anomaly-based sensor). The goal is to resolve contradictory information in different analyzers. Similarly, in [123] Svensson and Josang use subjective logic to reduce false alerts in the presence of uncertainty.

Data mining techniques are used in the works of Pietraszek [124][125] and Tian *et al.* [126] train classifiers. However, the necessity of human intervention to help on training becomes a complex factor to implement these approaches. Similarly, neural networks and fuzzy logic are used in Alshammari *et al.* [127]. This method also requires some training in order to be able to reduce false positives.

In Hooper [128], checking back hosts produces extra information about the probability that an alert is true or false. Host checking produces interesting information about the nature of alerts; on the other hand it introduces an additional level of complexity. It is not always the case that host checking is permitted by default, and if not the security policy of an organization may have to be significantly altered.

3.3.5. Alert Prioritization

Alert prioritization requirement is important in collaborative solutions because permit to prioritize the most critical alerts or class of alerts according to predefined metrics and severity and take appropriate actions for dealing with each one of them. Moreover, alert prioritization permit to enhance the alerts quality.

Normally, alert prioritization is focused on the detector's output and takes into account various domain information in addition to alert types or classes such as security policy, network topology, vulnerability analysis of the network services and installed software, and asset profiles.

The work of Yu *et al.* [91] focuses on the alert prioritization in two aspects: the not correspondence of the alert to any known attack (and consequently must be prioritized for further investigation) and the applicability of the attack against the protected network. Qui and Lee [113] propose an alert priority score calculated according to the severity and relevance of the attack. Porras *et al.* [129] propose an alert ranking based on the likelihood of the attack to succeed, the importance of the targeted asset, and the amount of interest in the type of attack. It is capable to evaluate alerts and clusters and is known as M-Correlator. The work of Alsubhi *et al.* [130] proposes a technique based on fuzzy logic for scoring and prioritization of alerts. In addition, a rescoring technique is also proposed to reduce the number of alerts.

Although these techniques have been show efficient in to evaluate alerts generated by IDS, they are not able to deal with ADS outputs due to the fact that a knowledge base is not exist.

3.3.6. Alert Prediction

Alert prediction is the ability to obtain a specific knowledge about determinate anomalies and attacks in order to anticipate and stop them before causing damage. The knowledge of previous anomalies and attack patterns is core to their prediction. One learns from history to avoid falling again in those traps previously encountered. By storing sequences that determine an attack signature, this may be identified and removed before taking place. Typically, alert prediction occurs in two phases. The first one is a training stage to enable the software to learn what characterizes an anomaly or an attack. The second phase is the comparison of partial patterns stored locally with the current alerts.

Over the years various approaches have been used to make alert prediction. Ye *et al.* [131] make use of EWMA (exponentially weighted moving average) forecasting method for intrusion detection, using a Markov chain model to learn and predict normal activities. A forecast of normal activities is used to detect a large deviation of the observed activities from the forecast as a possible intrusion into computer systems. A Chi-square distance metric is used to measure the standard deviation of a given activity from those considered normal ones.

Hu and Heywood [132] developed a two-stage attack prediction system (classification and prediction) aiming to investigate whether it is possible to predict attacks before they are initiated. This two-stage system employs Support Vector Machine (SVM) algorithm for classification and Self-Organizing Maps (SOM), a special unsupervised neural network technique, as predictor. Although interesting, the training and test results conducted using TCP connection feature from the DARPA KDD data set produced rates of 23.8% and 7.1% for false positive and false negative, respectively.

Wang *et al.* [133] introduced an alert correlation and prediction technique for multistage attacks. Based on the fact that the existing correlation methods use an in-memory index for fast searches and that finite memory is a limiting factor, they propose the use of a novel queue graph (QG) approach to represent an implicit correlation between new alerts and other alerts according to temporal order, allowing that alerts arbitrarily far away can be correlated. Moreover, a unified method based on QG approach was proposed to hypothesize missing alerts and consequently to predict future alerts at same time. Empirical results showed that this technique can process alerts faster than an IDS can report them, making of it a promising solution for an administrator to monitor the progress of intrusions.

Using the victim-end concept, Kannadiga and Zulkernine [134] developed an IPS called Event-based Network Intrusion Prediction System (E-NIPS). The idea of this work is to split an attack scenario into several stages depending on the actions taken during such incident. This way, attacks with similar or related goals are clustered in classes to reduce the processing of the prediction module. When the first stages of an attack is detected, i.e., when the initial stages of an attack corresponds to a class of attacks, alerts are released. The sequences of attack events are represented by rules, which are used to correlate detected attack classes in the attack scenarios.

3.4. Traffic Analysis

One of the shortcomings resulting from existing and traditional unwanted traffic solutions is the need of prior analysis, manually, to correctly detect unknown and undesirable traffic. This is not always feasible if it is considered the fast growing number of new applications and services. In addition, high-speed networks introduce non-trivial issues for these solutions including detection, characterization, and mitigation.

In this context, traffic analysis approaches have attracted special interest of researchers in recent years and presented promissory results against non-requested and undesirable traffic. Techniques to characterize the Internet traffic, methods to discover the traffic generated by applications, approaches to develop more accurate anomaly detection systems, and specific solutions to deal with specific types of unsolicited traffic (e.g., SPAM and P2P) have emerged in an attempt to facilitate the automatic, quick, and accurate identification and reduction of unwanted traffic.

This section describes some the most relevant works that have been proposed including its new methodologies, approaches, techniques, and algorithms for traffic analysis and that can be directly or indirectly applicable for unwanted traffic identification.

3.4.1. Non-Gaussian and Long Memory Statistical Characterizations for Internet Traffic with Anomalies

Scherrer *et al.* [135] propose a statistical model based on modeling aggregate traffic using time series and, as a consequence, offer an anomaly detection procedure based on such modeling. The authors argue that network traffic consists of IP packets arrival processes, which could be modeled using non stationary point processes or stationary Markov modulated point process but, due to the high volume of packets, these models would generate huge data sets. This way, they propose a model using marginal distributions and covariance functions, called the Gamma Auto-Regressive Fractionally Integrated Moving Average (ARFIMA) model.

The Gamma ARFIMA model [136] is defined as a stochastic stationary non-Gaussian long-range dependent process where the Gamma distribution solves the marginal problems and the ARFIMA covariance function deals with long-range dependence (LRD) processes. This model is assumed to be 2nd order stationary. The Gamma distribution (first order stationary) satisfactorily models traffic marginal distributions for both small and large range scales of aggregated traffic (at byte, packet or flow), as opposed to a Gaussian distribution like log-normal and Weibull. The ARFIMA covariance function (defined as being second order stationary) is a natural choice because it allows dealing with both short and long-range dependence. The Gamma ARFIMA model uses only five parameters (α and β from the Gamma distribution; ϕ , d , and θ for ARFIMA), adjusted according to each aggregation level. The preliminary results indicate a good adequacy in both small and long aggregation levels. Moreover, the use of these parameters provides a simple, highly flexible and practical solution.

In addition to the Gamma ARFIMA model, the authors also describe a scheme to detect anomalous traffic including legitimate (Flash Crowds) and illegitimate (DDoS attacks) traffic. The adopted behavior pattern is generated through the analysis of the

statistical profile at various time-scales of the series, which has been shown to be sensitive to changes caused by anomalies. Hence, what is needed is to explore the multi-resolution nature of the problem. Consequently, the scheme works by splitting each time series during analysis into adjacent non overlapping time windows (one minute, initially) and computing, for each time window, a distance (Mean Quadratic Distance - MQD) between a statistical characteristic measured during the window and the same one measured on an a priori chosen reference window. In order to detect anomalies, thresholds are applicable for the calculated distance where an unexpectedly large deviation is assumed to signal anomalous traffic behavior. The initial results of this procedure detection are encouraging indeed when taking into account the high hit ratio for both large and even low rate illegitimate traffic. The authors also explain the necessity to explore the analysis over several minutes as a mean to increase the degree of detection.

In order to evaluate the proposed analysis, the authors conducted some experiments using standard data from major available Internet trace repositories such as PAUG and LBL-TCP-3 [137], AUCK-IV [138], CAIDA [139], and UNC/FORTH [140] and time series collected from the RENATER [141] network.

- **Advantages:** generally speaking, the idea presented in this work, despite not being a new one, could give anomaly detection a great step forward for many reasons. First, unlike other works, this is a simple model that is capable of representing Internet traffic using only five parameters. Second, its versatility in that it can portray traffic behavior at different aggregation levels could be used to create more efficient and realistic traffic generators. Third, the results of procedure detection are encouraging indeed when taking into account the high hit ratio for both large, and more importantly, low rate illegitimate traffic (denial-of-service attacks).
- **Limitations:** the hit ratio of the detection procedure for flash crowds can sometimes be very small and remained below 15% in most evaluated scenarios. Moreover, the estimation of ARFIMA covariance and model parameters is hardly trivial but nonetheless feasible.

3.4.2. Extracting Hidden Anomalies using Sketch and Non Gaussian Multiresolution Statistical Detection Procedures

Dewaele *et al.* [142] introduce a procedure especially tailored for detecting hidden low-intensity anomalies in Internet traffic. This technique combines sketches and a non-Gaussian statistical model to discover anomalies in traffic data. Sketches permit to reduce the data dimensionality and measure the reference traffic behavior, whereas the Gamma distribution extracts the shape parameter of the marginal distributions of the traffic for each individual sketch and each aggregation level, and accurately captures short-time correlation structures of traffic [136]. The detection procedure makes use of the Mahalanobis distance [143], a statistical measure used to determine similarity of an unknown sample set to a known one, to perform comparisons among sketches, and is thus to determine anomalous behavior.

This detection procedure consists of the following main steps: sketches generation, multi-resolution aggregation, non-Gaussian modeling, reference, statistical distances, and anomaly detection by sketches combination.

1. **Sketches generation:** sketches or random projections are used to divide the traffic packets into sub-groups according to sliding time-windows. For each one of these time slots, only the arrival timestamp, source and destination IP address information and port numbers are analyzed. As a result, hash tables are then generated, representing sub-traces of the original trace, while using IP source or destination addresses as their hash keys. Figure 3.1 exemplifies this process.

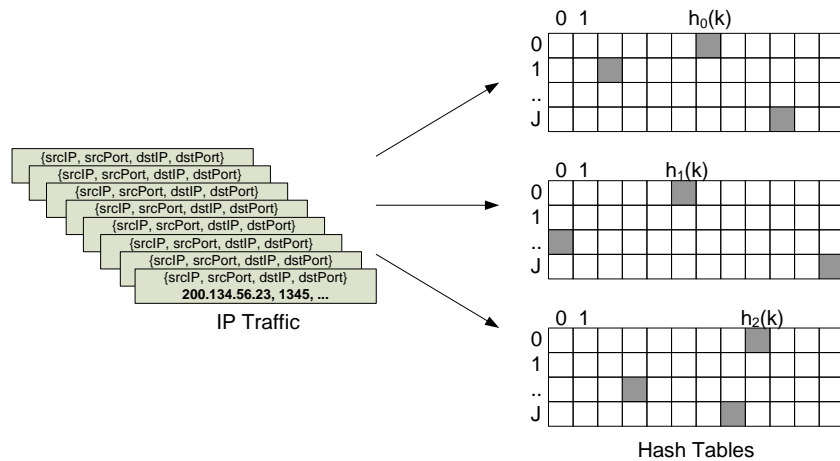


FIGURE 3.1: Sketches generation process

2. **Multiresolution aggregation:** the sub-traces are next put together to form aggregated time series for a specific scale.
3. **Non Gaussian modeling:** the Gamma distribution is used to describe the marginal distributions of aggregated traffic time series. In other words, for each aggregated hashed time series of an aggregate time scale, the Gamma parameters (α and β) are estimated for use to calculate reference and statistical distance. The Gamma distribution choice and adequacy are explained in [135][136][144].
4. **Reference:** the average behaviors and typical variability are estimated for each hash using mean and variance estimator. In spite of simplicity, the joining of sketches and reference permits the definition of normal and anomalous behavior patterns (anomalies can be found observing changes in statistical patterns comparing sketches with others at the same time).
5. **Statistical distance:** the Mahalanobis distance is used to measure anomalous behaviors of references. Each calculated distance is matched against a threshold. If the distance of reference is less or equal to a threshold, it is considered normal, otherwise it is classified as anomalous.
6. **Anomaly detection by sketches combination:** the anomaly detection is realized comparing sketches (hash keys) with attributes (source and destination IP address, and port numbers) registered in a list during the detection process.

All the process is illustrated by Figure 3.2.

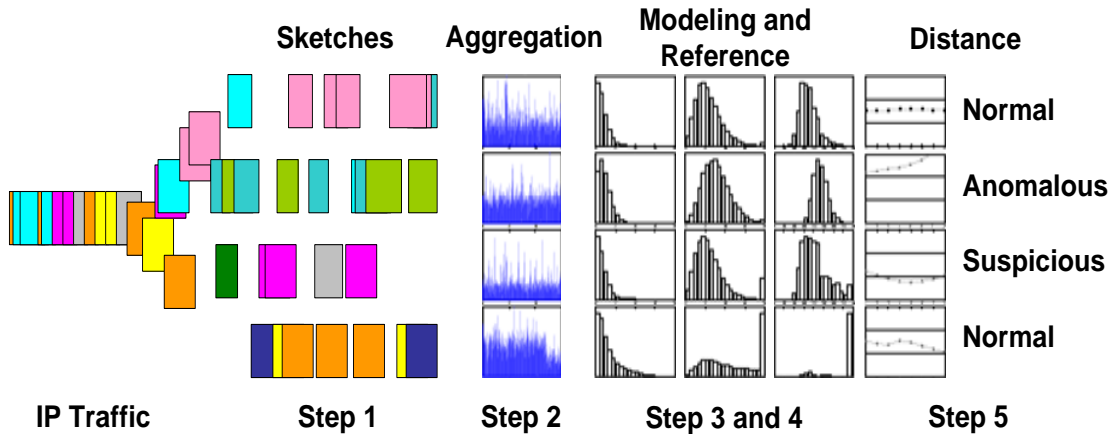


FIGURE 3.2: Anomaly detection stage

The authors deployed and validated this strategy using data from the MAWI traffic repository [145] through two case studies: one with low-intensive long-lasting spoofed flooding and another one with a short-lived port-scan. The results demonstrate that the procedure is able to discover elephant (large) packets, flash crowds, DDoS attacks (SYN and ICMP flooding), scan activities, P2P traffic, and worms, among other anomalies.

- **Advantages:** in spite of this strategy being a work in progress, the initial results are very promising. First, no prior knowledge of the traffic and its characteristics is necessary. Second, the procedure seems to be capable of detecting short-lived anomalies as well as longer ones. Third, it requires very low computational power and can therefore be implemented in real-time. Lastly, its “detection window” can work at less than one minute as well as over longer than a minute period.
- **Limitations:** as raised by the authors, sometimes, the legitimate traffic (DNS, for instance) could be recognized as illegitimate if it presents a unique traffic pattern. Hence the authors suggest the addition of filters to exclude known patterns during the analysis phase. Note that attackers knowing the use of such filters may use them to hide their attacks.

3.4.3. A Novel Approach for Anomaly Detection over High-Speed Networks

Salem *et al.* [146] propose a new framework for anomaly detection for use over high-speed links. It provides an early and efficient detection as well as the appropriate and corrective countermeasures. Unlike others, the proposed approach does not depend on any traffic distribution parameters and their variations, and does not suffer from a possible lack of capability for handling large state space generated by traffic information at high-speed links. To achieve this, it uses a new variation of sketches to aggregate multiple data streams and a parametric version of the multi-channel cumulative sum (M-CUSUM) algorithm to detect anomalies in each sketch.

The methodology is organized in two steps. Firstly, Count-Min Sketch (CMS), which permits random aggregation of flows without any significant disruptions from their variations, is employed to store large traffic volumes in flows. This is achieved with a small amount of memory and with a little complexity degree, for a fixed time interval T . Secondly, M-CUSUM is applied to each sketch. As a result, it is then

possible to identify the keys of the sketches that are mapped with a raised alarm by CUSUM. However, before being in a position to identify any anomalies yet, the authors needed to solve some relevant issues. First, they argue that sketches, in general, are not reversible, i.e., they do not preserve the key (e.g., source IP address) of the flows, and when anomalies are detected, it is difficult to infer the culprit flows. The only solution is to test all possible entries, by hashing these for the second time, to match that consist anomalous flows. As this solution is neither scalable nor desirable, they propose a new variant for the sketch by adding an inversion procedure to it.

The idea is based on exploring a new index in an additional sketch named Multi-Layer Reversible Sketch (MLRS), where such index is used to store keys. MLRS is used in the same way as the initial CMS sketch, where the arrival of any key increments its associated counter. However, each key has l counters (one for each layer), and the key of N bits is split into $l \times w_0$ bit, with $w_0 = 2^P$, and $l = \lceil N/P \rceil$. P is the number of bits used to split the key, and w_0 is used as layer width (number of columns) in MLRS [146]. Since anomalous flows must have one alarmed bucket in each layer and this methodology executes hierarchical searching, if a search does not find at least one bucket raised alarm by CUSUM in each of i^{th} ($i \leq l - i$) first layers of MLRS, there is no need to continue searching in other deep layers.

On the other hand, if there is at most one anomalous flow in each layer, the suspect key is obtained by the concatenation of the l indices in MLRS. However, it is necessary to ensure the validity of the candidate key. At this stage a new challenge is raised: that of collision with other IP prefixes. Due to the large amount of traffic information, many buckets in different layers may be subject to possible collisions, which in some cases will generate a bigger set of keys to verify than the originally determined. The suggested solution was in using an IP mangling technique [147]. This is a reversible procedure that randomizes the input data in an attempt to reduce and destroy correlation between keys. It is based on an optimized version of RC4 (Ron's Code) [148] ciphering algorithm. The confirmation makes use of a query of CUSUM functionality in CMS.

The authors evaluated the proposed framework using many public traces including LBL-TCP-3, Abilene, Auckland, and within OSCAR RNRT French Research project. The implementation used Endace DAG 3.6ET Gigabit network interface [149] sniffers.

- **Advantages:** this proposed framework combines sketches with M-CUSUM to develop a powerful tool for detecting traffic anomalies. One that especially considers DoS and DDoS attacks even when exhibiting low rates over high-speed links. This approach is flexible and could be easily decentralized. The timescale of detection can be reduced to as little as a minute or even less.
- **Limitations:** M-CUSUM raises alarms only at the starting phase, expected to take few minutes when subjected to constant rate attacks. In addition, the proposed methodology needs to be adjusted to work with other strategies such as those that deal with anomalies detected through the examination of TCP flags and other protocols.

3.4.4. Anomaly Detection of Network Traffic based on Wavelet Packet

Gao *et al.* [150] describe a new network anomaly detection method based on wavelet packet transforms. The authors argue that there are four problems seen in current methods based on wavelet transforms when used to detect anomalies. First, almost all of these use multi-resolution analysis, considered only adequate for detecting low frequency anomalies. Second, their results can be incorrect or misleading when only a single scale is analyzed. Third, wavelet transforms demand relatively high computational power and, consequently, can be considered inappropriate for real time operation. Lastly, choosing the adequate time-windows and calibrating their respective thresholds remains a challenging exercise.

As a workaround these limitations, they propose the use of wavelet packet analysis, which is capable of decomposing a signal, hence offering a diverse range of possibilities for analysis. Overall, wavelet packet analysis permits that a signal can be split into an approximation and details parts. In turn, the approximation is then itself split into a second-level approximation and details, and the process is repeated many times over again. For n -level decomposition, there are $n+1$ possible ways to decompose or encode the signal.

The proposed detection method is able to adjust the decomposition process adaptively, and exhibit the same detection ability across low, middle, and high frequency anomalies. In order to achieve this objective, the authors use one fast wavelet packet algorithm based on sliding window aiming to decrease the nature of computation complexity reminiscent from wavelet packet transforms, and a statistical detection algorithm based on scores together with a thresholds based mechanism to discover anomalies. The detection process employs an initial anomaly detection stage, to verify at each scale by means of a statistical detection algorithm whether there are anomalies. This is achieved by analyzing wavelet packet coefficients (representations of signals present in the wavelet transform). In the case that an anomaly has been perceived, a new decomposition of wavelet packet coefficients is made and this step is executed again. The decomposition levels are self-adaptive. In the event that there is an anomaly, the reconstruction of wavelet packet and confirmation of anomaly stages are used to reconstruct the signals from the initial scales and check whether the reconstructed signal is anomalous, respectively. The main goal here is to reduce the number of false alarms.

To validate their approach, the authors simulated a number of scenarios using LBL trace as background traffic and the public domain Network Simulator software (NS-2) to generate anomalous traffic (DDoS attacks).

- **Advantages:** the simulation results have demonstrated that the wavelet packet analysis is a promissory technology for anomaly detection.
- **Limitations:** this procedure only was tested to discover DDoS attacks. Moreover, the time spent in the detection process increases with the number of signal decompositions. Last, the representativeness of the generated DDoS attacks remains to be proven. An alternative would be to include other independently generated DDoS traces into the simulated scenarios.

3.4.5. Profiling Internet backbone Traffic: Behavior Models and Applications

The methodology proposed by Xu *et al.* [151] and aims to identify traffic anomalies by profiling Internet backbone traffic. The profiling method uses data-mining and information-theoretic techniques to extract and classify flows. It automatically discovers significant behavior patterns from link-level traffic data, and to provide plausible interpretation for the observed behaviors. In particular, this approach places great emphasis on the entropy concept. Entropy can be seen as the measurement of information of a given dataset, which essentially quantifies “the amount of uncertainty” contained in that dataset [152].

This methodology uses different metrics to calculate the entropy such as the quantities of flows and bytes. The profiling methodology implements an identification technique, which consist of four related basic stages:

1. **Preprocessing:** deals with the definition of the domain data that will be processed. In other words, in this stage the packets are captured and aggregated into flows.
2. **Extracting significant clusters:** determines the clusters for four features or dimensions. This procedure aims to reduce and facilitate dataset behavior inspection through the identification of its most significant or principal elements. The extraction of significant clusters deals with a four dimensional feature space composed by the four attributes srcIP, dstIP, srcPrt and dstPrt. Considering these elements, it is possible to identify two relevant types of network communication behavior. Firstly, there is a relationship between IP addresses (srcIP and dstIP), one that determines the communication pattern between hosts. Secondly, there is also the behavior built from port/service (srcPrt and dstPrt) usage patterns.
3. **Clusters classification:** classifies each cluster’s element into behavior classes based on similarities and dissimilarities of communication patterns (ports and IP addresses).
4. **Communication patterns interpretation:** defines a set of behavior classes capable of better describing given applications and services.

Figure 3.3 shows the interconnection among the four stages and how they are fed and interact. This process allows an automated or supervised adaptation of parameters. For instance, the iterations allow that information coming from communication patterns interpretation to affect the decisions taken on preprocess stage.

In the profiling methodology, entropy is used to measure the amount of relative uncertainty (RU) contained in the significant clusters extracted from a fixed dimension (e.g. source IP). Next, behavior classification based on communication patterns of end-hosts and services are made. Therefore, for every cluster, an RU is computed and used as a metric to create behavior classes (BC). Among these classes, it should then be possible to identify which one represents anomalous traffic. As result, the study presented in [17] have proven their ability to detect a wide variety of massive anomalies such as port and IP scans, DoS and DDoS attacks, among others.

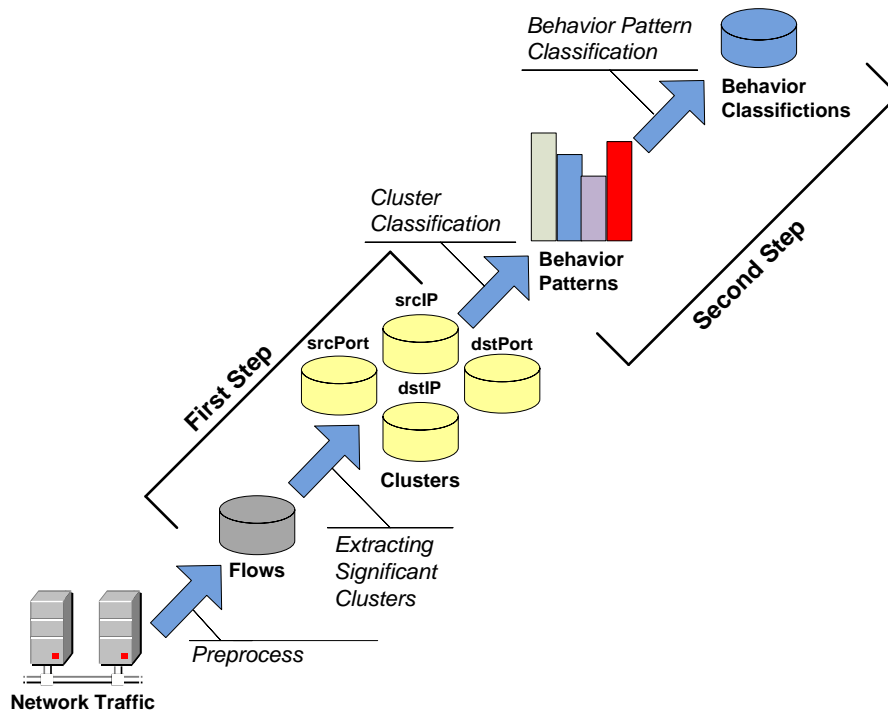


FIGURE 3.3: Profiling Methodology

The profiling approach was implemented and validated using datasets composed by packet headers (only the first 44 bytes for legal and privacy reasons) collected from different links of a large undisclosed ISP. These traces varied in their duration between 3 to 24 hours whereas the capacity of the monitored links ranged between 155Mbps to 10Gbps.

- **Advantages:** traffic profiling is seen as a powerful tool for network operators and security analysts with applications to critical problems such as detecting anomalies or the spread of hitherto unknown security exploits, profiling unwanted traffic, tracking the growth of new services or applications. Moreover, this approach is flexible and capable of automatically discovering others significant behavior patterns.
- **Limitations:** the profiling methodology is appropriate to a single backbone link, not network-wide traffic. Moreover, flow analysis is not adequate to detect low-intensity attacks. Lastly, the process of packet header collection and posterior aggregation in flows in real-time is time and CPU expensive, especially when considering emerging multi-gigabit broadband interfaces. The authors in [17] draw the attention to the fact that the minimum time needed to transform collected packets into flow-level statistics is five minutes. This cannot be acceptable in the context of real-time network and service security. A further limitation of traffic profiling is that it is useful for identifying traffic profile classes only and fails to distinguish between different applications that exhibit similar profiles. Profiling may therefore need some other complementing strategies to work effectively.

3.4.6. Mining Anomalies Using Traffic Feature Distributions

Lahkina *et al.* [153] propose an anomaly detection methodology based on the distributions of packet features capable of identifying low and high volume anomalies.

The authors argue that Origin-Destination (OD) flow analysis can reveal a diverse and general set of anomalies, especially the malicious ones. The proposed method uses entropy to make observations and extract very useful information with regard to dispersions in traffic distribution. The mining methodology is organized in two main phases: traffic feature distributions and diagnosing methodology.

During the former, traffic feature distribution, extracts fields from packet headers to look for anomalies possibly caused by changes (dispersions) in the distribution addresses or ports being observed. For example, during a port scan event, the distribution of destination ports will be more dispersed than during normal operations. Four IP header fields were examined: source and destination IP addresses and source and destination ports (srcIP, dstIP, srcPort, and dstPort). The authors claim that it is possible to capture the degree of dispersal or concentration of a distribution using entropy, since common anomalies, for example port scans, can viewed clearly in terms of entropy in comparison with traffic volume.

Next, the results of the entropy application phase are then fed to a second phase where a distribution is used to reach a diagnosis and classify anomalies. Consequently, the multiway subspace method is used to detect anomalies and offers an unsupervised classification strategy to classify them. This method is a derivate of the subspace method proposed in [154], and which results in traffic analysis study in [155]. The idea behind this method is to identify correlated variations on multiple traffic features (srcIP, dstIP, srcPort, and dstPort). Such correlations should point to possible anomalies. The unsupervised classification uses a clustering approach to form clusters, where the data is then analyzed to discover anomalies. It also occurs in two phases. First, known anomalies are clustered in order to gain knowledge of how anomalies emerge. Thus, the clusters are labeled based on their types. Next, the classification is performed by clustering unknown anomalies.

Datasets collected from both the Abilene [156] and Géant [157] research backbone networks were used. The results show that the prototype was able to identify a wide range of anomalies including alpha flows, DoS and DDoS attacks, flash crowds, varied scans (port, IP, network, worm), outages, and some unknown anomalies.

- **Advantages:** the use of entropy helps detecting variations on network traffic caused by the most diverse anomalies. The results obtained in the traces from high-speed backbone prove that entropy could be more efficient than volume-based techniques. Moreover, the proposed multiway subspace method also demonstrates its superior handling of huge volumes of OD flows and, consequently, to discover anomalies.
- **Limitations:** the implementation complexity and the necessary computational power are pointed out as limitations of this methodology. Moreover, the time necessary to build OD flow time series is large as it is in the order of few minutes. Similarly, the method has been validated using academic research networks. Commercial traffic tends to exhibit different profiles as the applications and speed of their adoption differ between the two scenarios.

ChkModel

Aschoff [158] proposes an anomaly detection methodology based on statistical analysis of TCP connections behavior capable to identifying low and high volume of TCP

attacks. ChkModel is designed to distinguish between well and badly intentioned traffic, and also to identify possible service resource problems. It observes the total traffic between clients and servers, at *connection* and *socks* granularity. A *connection* is represented by the combination of the following elements: IP addresses and port numbers of the clients and servers. A *sock* is defined as a collection of connections that have the same IP address and port number of the server [159]. Connection classification is used to create the good client list, and socks classification is used to detect attacks or resource problems. Figure 3.4 illustrates the concepts of a connection and a sock.

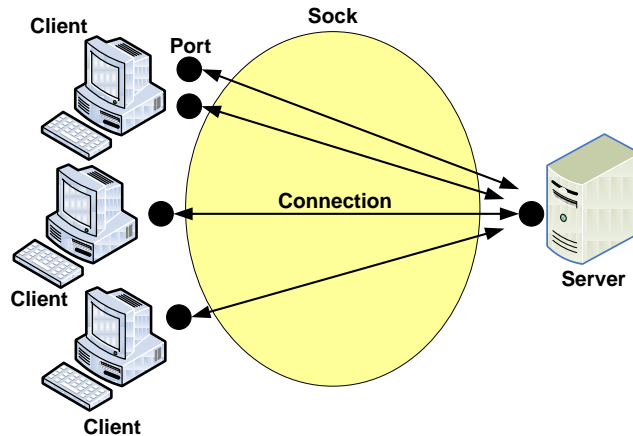


FIGURE 3.4: ChkModel Sock and Flow Schema

Figure 3.5 depicts a ChkModel deployment at the network edge. Rather than establishing the legitimacy of individual packets, ChkModel observes connection and sock behavior and classifies them as being legitimate or attacks. As shown in Figure 3.4, the ChkModel is basically composed of the traffic observation and classification components. The traffic observation component is responsible for capturing and aggregating packets based on connections and socks. These are then fed into the classification component responsible for identifying any anomalous behavior based on known legitimate connection and socks models.

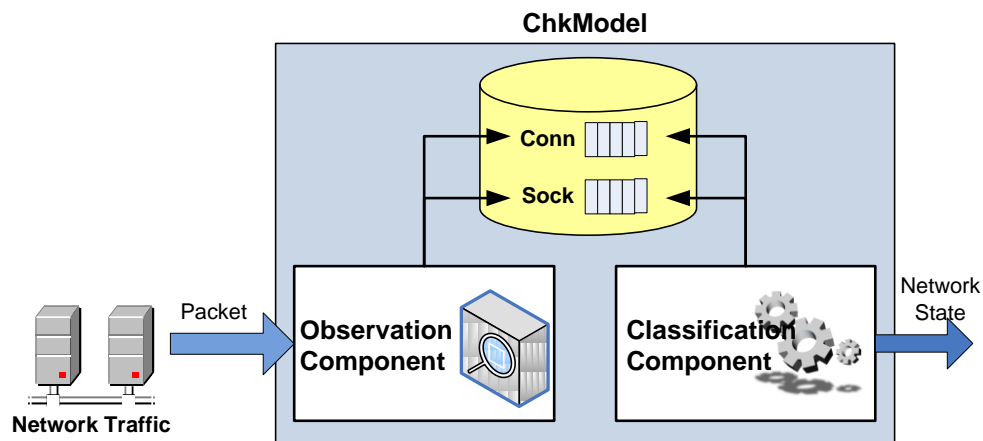


FIGURE 3.5: ChkModel Architecture

The traffic observation component is responsible for collecting real traffic at one or more points on a target network and for gathering traffic statistics at connection and sock granularity which are then stored in the *Conn-Hash* and the *Sock-Hash* tables respectively. These statistics are continuously read by the classification component that compares them with existing legitimate connection and sock models. The current

implementation of the ChkModel verifies only the legitimacy of TCP traffic, which currently reaches as much as 90% of traffic in the Internet [158]. Judging a TCP connection as legitimate is based on the two-way communication paradigm of the TCP protocol. A TCP connection is modeled by the ratio of the number of packets sent to and received from a specific destination [159]. This relationship will be used as a baseline for the implementation of legitimate connection and sock models, as described in more details next.

Legitimate TCP Connection Model

Data flows from the source to destination during a TCP session are controlled by the constant flow of acknowledgments in the reverse direction. Congestion can be perceived if the flows of acknowledgments decrease. In this situation, TCP reduces the sending rate. This explains why normal TCP communication can be modeled by the ratio of the number of packets sent to and received from a specific destination. A legitimate TCP connection model defines two thresholds, *ConnThr1* and *ConnThr2* as the maximum ratio (number of packets sent to or received from) for a healthy TCP connection and the maximum ratio for a suspicious connection respectively. ChkModel classifies a connection as *good* if its ratio is below *ConnThr1*, as *suspicious* if its ratio is between *ConnThr1* and *ConnThr2* otherwise it is classified as being definitely bad. Because of the dynamic nature of network behavior, some normal traffic may sometimes be misclassified as bad. In order to deal with this problem, a second threshold which provides a new classification, called suspicious, for borderline cases for use in the sensor was created.

Legitimate TCP Sock Model

ChkModel's legitimate TCP sock model is similar to the previous one. The main difference is that it defines only one threshold namely *SockThr*, seen as the maximum ratio for a normal sock. Recall that sock is defined as a collection of connections that have the same IP address and port number of the server. A sock is classified as normal if its ratio is below *SockThr* and as an attack otherwise.

When a TCP sock is low, it is an indication that a determined server is overloaded, possibly by an attack, and through the TCP connection model, the presence of an aggressive sending host in the *Conn-Hash* Table signals the possibility that a host can be participating in an attack.

The ChkModel methodology was written in C++ and validated using a GPRT testbed.

- **Advantages:** the use of a statistical function to model TCP connection permits to detect the most diverse anomalies TCP related. The results obtained on initial tests prove that it can be used in both on-line and off-line traffic monitoring. Moreover, the use of adaptive thresholds helps to adjust to the real network state. Last, this is appropriate both for a single backbone link and for network-wide traffic.
- **Limitations:** this methodology was designed to discover only TCP attacks. Moreover, as raised by the authors, sometimes, illegitimate traffic (SPAM, for instance) could be recognized as legitimate if the target (server) was capable to respond great part of connection requests.

3.4.7. Discussion of evaluated traffic analysis approaches

In spite of the advances in unwanted traffic detection, especially over challenging high-speed links, these approaches either present a costly price of computational complexity and infrastructure changes or some type of inadequacy related with some features such as timescale of detection and detection range.

For instance, the behavior-based techniques Profiling [151] and Mining [153] are very similar but diverge over some aspects. In terms of computational complexity, the first one is simpler. On the other hand, whereas Mining perceives both low and high intensity anomalies, Profiling is only adequate to massive anomaly detection. Other good examples are statistical approaches [136] and [142]. Both are conceptually similar, but they differ in their ability to detect a great number of anomalies and the timescale. Whereas [136] has a poor performance in detecting two types of anomalies, [142] reaches a varied scale of anomalies. On the other hand, the timescale of detection in [142] can be more than fifteen minutes whereas in [136] it is less than one minute.

Finally, while by no means comprehensive, this work is of the opinion that this chapter captures the essence of the discussion and analysis about anomaly detection techniques. Table 3.1 summarizes the comparisons of anomaly detection techniques presented.

TABLE 3.1: A subjective comparison of various anomaly detection techniques

Approach	Metrics			
	<i>Analysis Technique</i>	<i>Data Format</i>	<i>Time of Detection</i>	<i>Anomalies Identified</i>
Non-Gaussian Multiresolution	Statistical analysis	Aggregated traffic	At least one minute	DoS and Flash crowds
Sketches and Multiresolution	Statistical references	Sketches	For low-rate, less than one minute. For high-rate, until 15 minutes	Flash crowds, DDoS, Scan, and unwanted traffic like P2P and worms
Sketches and CUSUM	Parametric M-CUSUM	Sketches	At least one minute	DoS and DDoS attacks
Wavelet	Wavelet Packet Analysis	Not applicable	Not defined	Massive and unknown attacks
Profiling	Data mining and Entropy	Flows	At least five minutes	Port and IP scans, DDoS and unknown attacks
Mining	Statistical multiway subspace	OD flow	At least five minutes	Alpha flows, Flash crowds, scans, DDoS and unknown attacks
ChkModel	Statistical analysis	Connections and Socks	Instantly	TCP attacks

3.5. Chapter Summary

This chapter introduced an overview of solutions against unwanted Internet traffic. First, a brief discussion about traditional and promissory approaches was presented. Common filtering mechanisms (including firewalls, ACLs, proxies, and BGP null routing), IDS, anti-something software, and honeypot were showed and its usefulness and shortcomings in to deal with unwanted traffic were pointed out. After, the recent and promissory solutions (advanced filtering mechanisms and IP space investigation) have also been presented in this section and examples of how they can be used to deal with unwanted traffic were showed.

Next, collaborative solutions (CIDS and CAIDS) were presented and discussed. Overall, the most important researches and contributions in this area were shown as soon as the essential requirements for building such solutions were contextualized.

Lastly, an overview of relevant research in traffic analysis to detect unwanted traffic was provided. The state of art in techniques used for detecting unwanted traffic was surveyed. The review was organized with the goal of achieve a great spectrum of anomalies. Solutions based on statistical distributions, mathematical models, and, mainly, behavior analysis of hosts and network were studied and, then their advantages and coverage and also their limitations were discussed.

To sum up, the aim of this chapter is not claim that all of the existing strategies are reviewed here but it should give the reader ideas and argue for the need to continue such efforts as a solution remains a further undertaking. As practical result, this chapter demonstrated the fact that no single or isolated strategy is sufficient to deal with unwanted traffic. The main reason is because it is a very complex and dynamic. Despite initiatives like the IETF to provide some general guidelines (BCP 38 and BCP 84) for operators and providers to put in place common procedures and efforts to root out unwanted traffic and save on the resources, new attack forms keep popping all the time, for different purposes and targeting different services. A solution that may achieve some reasonable results would only be the result of cooperating processes, administrators, security alert sites and any other mean that could contribute to this endeavor. This thesis considers collaborative and cooperative solutions as its focus and a significant step in the direction self-defending networks. In the next chapters, the proposed approaches for analyzing and diagnosing unwanted traffic and anomalous will be introduced.

Part II
OADS Approach and Tools

Chapter 4

Orchestration Anomaly Detection System (OADS)

Internet services relied for a long time on the informal agreements and goodwill of member sites in protecting and correctly forwarding each other's traffic. Although lacking a centralized control or ownership, the Internet is one of the few, if not the only, self-governing infrastructure that manages to operate reasonably well under such a paradigm.

Today, that trust model is coming under intense attacks as a result of the diversified communities that joined the Internet bandwagon. Figure 4.1 illustrates an example (an email received in nanog.org list) of the almost routine discomfort and trouble caused by unwanted traffic.

<p>Date: Mon, 15 Sep 2008 08:14:06 -0400 (EDT) From: Tom Obama <tb@dyndns.com> To: nanog@nanog.org Subject: Paging Level(3) Security Operations</p>
<p>Hello NANOG list, I'm trying to reach out to Level(3) Security Operations for assistance with a Denial of Service attack. So far, the normal means to contact Level(3) have failed.</p> <p>I can be reached directly at 679-798-1248.</p> <p>Thanks, Tom Obama</p>

FIGURE 4.1: E-mail example of the fight against unwanted traffic

Can the Internet way of life be maintained? For how long and at what price? What can be done to make a network administrator's work easier to handle such problems? Would not be nice to have some automatic, quicker and highly efficient response to similar unwanted traffic? One cannot simply blacklist the domains where unwanted traffic comes from, as this would only benefit those who are exploiting existing Internet weaknesses or preferably the choice of a simple design.

In an attempt to answer to such concerns this chapter presents the specification of a new orchestration-based approach to detect, and as far as possible, to limit anomalous (unwanted) traffic. Core to the proposal, is a framework that coordinates the receiving of a multitude of alerts and events from detectors, evaluates this input to detect or prove the existence of anomalies, and last choose the best action to be taken, named **Orchestration-oriented Anomaly Detection System (OADS)**.

In order to explain how this proposed approach could be useful in improving the anomaly detection process, firstly, an overview will be made aiming to explain the orchestration concept and how it is applied in this proposal. Next, all OADS

architectural components will be individually detailed. Lastly and foremost, the benefits of the proposed approach will be explained, detailing how it will deal with special services and comparing it with some existing specialized architectures.

4.1. OADS Overview

The research in the field of collaborative anomaly and intrusion detection systems is extensive and currently, but yet only very few systems have been simulated or implemented. In addition, due to the technological trends and evolution of strategies and mechanisms to generate unwanted Internet traffic (anomalous traffic), the typical collaboration solutions can be considered deficient to deal with the current level of Internet traffic, especially for high-speed links. Aiming to address these issues, a new approach based on the concept of orchestration services is proposed called **Orchestration oriented Anomaly Detection System (OADS)**.

Orchestration is nothing but a modern metaphor that describes an already well-known security network administrator activity. Analogously to a musical concert maestro responsible for keeping the rhythm, cuing the different players, security managers organize the harmony and rhythm of various detection instruments (IDS, ADS, remediation systems, firewall, walled gardens, traffic analysis appliances, and so on) to achieve a desired effect, turn the network as more secure as possible.

The idea behind OADS approach is to automatically manage the execution of different anomaly detectors traditionally unaware of each other's. In other words, the proposed approach permits and explores the added benefits obtained from the collaboration and harmonization among different techniques against malicious activities. Collaboration enables two or more processes to work together towards a common goal without the need for a pre-established leadership. In the music world, this occurs when musicians work on the same musical album or individual song. In the information security context, this thesis sees collaboration as a facilitator of relationships between different anomaly detectors. For instance, two or more detectors can share the same traffic base (traces) to perform analysis or the final result from one that may be taken as input for another detector to help reach a better decision. *Harmony*, also seen as an interesting concept, means that two or more different sound notes fit well together. Extending such concept to the network security area, one can say that harmonization is enabling a service from any source, exposed through any technology, to work well with an orchestration.

4.2. OADS architecture

To better clarify the orchestration approach, Figure 4.2 illustrates its organization and shows its components.

The OADS architecture consists of four basic elements: Alert Pre-Processor, OADS Analyzer, Decision Service, and OADS Miner. In addition, OADS approach uses anomaly detectors as external elements. This section explains in details each one of them.

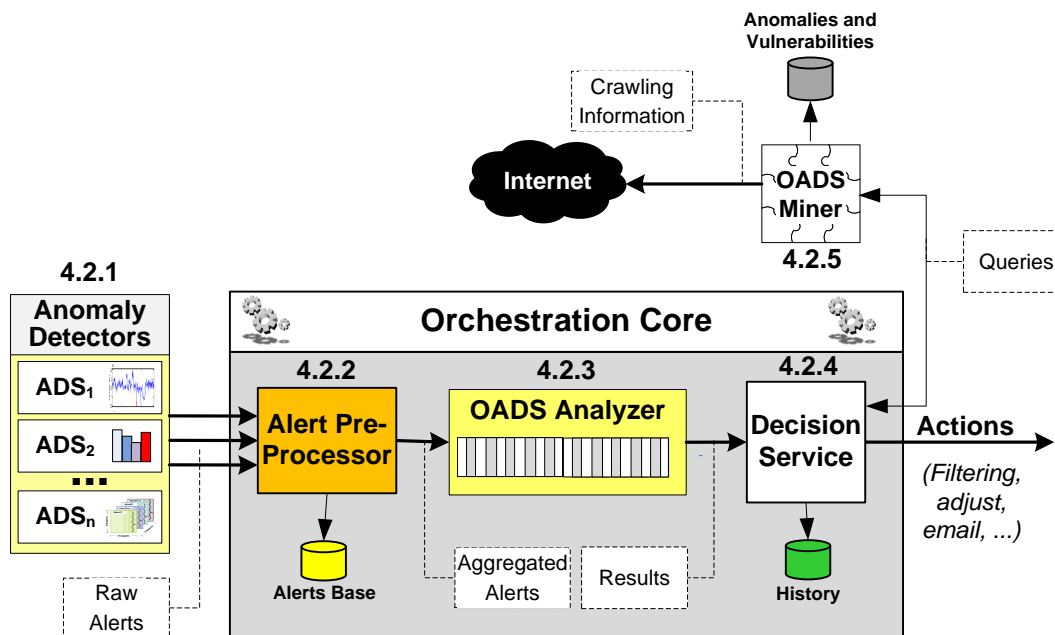


FIGURE 4.2: OADS architecture view

4.2.1. Anomaly Detectors

Anomaly detectors implement the intelligence necessary to analyze traffic information from its sensors looking for potential attacks or abnormal events. They extract data related to suspicious and anomalous traffic and supply the orchestration core of OADS approach with alerts. In other words, the main goal of an anomaly detector is to identify and notify the likely existence of any anomalous, unwanted or harmful traffic behavior to the orchestration core.

In practical terms, anomaly detectors are logical devices that may be implemented in hardware or software. In hardware, they are commonly aggregated with sensors that implement a variety of techniques such as sampling and filtering, packet level capture, flow aggregation, and DPI. Currently, there are a great number of hardware-based products to capture and inspect network traffic in real time that has been developed. P-Series (Force 10 networks) [160], Orca-flow (Cetacea networks) [161], and Cloudshield technologies [162] are some examples. Regarding software-based, a plurality of solutions, tools, techniques, and systems to process the traffic may be used, including IDSs (Snort [163], Bro [164] and Prelude IDS [165]), honeypots (Honeyd [75] and Nepenthes [166]), and academic prototype software or solutions [151][167].

4.2.2. Alert Pre-Processor

Alert Pre-Processor component can be seen as the front door of the OADS approach. It plays a role that consists of receiving information (raw alerts) from anomaly detectors and preparing them to be analyzed. Basically, it performs two activities.

The first one is the **adequacy** of alerts. Although OADS approach adopts IDMEF standard as alert message format, is necessary a content adequacy since some anomaly detectors present distortions in relation to IDMEF output, as previously presented in Section 3.3.1. A good example involves two famous free IDS software:

Snort [163] and Prelude IDS [165]. Basically, they have different nomenclatures for alert identification field. Figure 4.3 highlights this divergence.

<pre><?xml version="1.0"?> <IDMEF-Message version="1.0"> <Alert <i>messageid</i>="3004"></pre> <p style="text-align: center;">(a) Snort IDS</p>	<pre><?xml version="1.0"?> <IDMEF-Message version="1.0"> <Alert <i>ident</i>="abc123456789"></pre> <p style="text-align: center;">(b) Prelude IDS</p>
---	---

FIGURE 4.3: Snort and Prelude nomenclature problem.

While Snort employs the standard nomenclature (*Alert messageid*), Prelude names this field as *Alert ident*.

The second is **aggregation**. Since attacks and anomalies might consist of one or multiple steps and the anomaly detectors are capable to create alerts for each of these steps, aggregation activity makes easy to build hypothesis about the anomalies and possible defense strategies, and to reduce the volume of data.

Alert Pre-Processor component may employ two aggregation schemes: similarity and cluster. The first and most used scheme explores the distance in time between alerts with some similarity between determined alert fields. The idea behind it is to aggregate “near” alerts, that is, to fuse alerts if they are both close in time and typical attributes such as Source IP, Source Port, Destination IP, Destination Port, Attack Class, and source detector are similar. This thesis does not make use of Port fields since source ports can be easily changed to hide the attack and destination ports are normally related to attack class field. This aggregation approach is very simple and must be utilized to evaluate a reduced alert number, i.e., it is adequate for simple attack scenarios.

The second is based on clustering, where alerts set are divided into different groups. The goal is to group alerts that have the same attack scenario. Since the majority of attacks are against the same target machine (or other network devices), so the attacks with the same target IP, for example, often have greater similarity. Consequently, it is easier to find similar alerts. The choice for this aggregation approach is justified by the need of working with massive alerts, especially when checking high-speed links.

In addition, in order to provide security requirements, the Alert Pre-Processor component can implement two security measures: session control and cryptography. The former is to register all participants and consequently enhance the general security of the OADS approach. Universal Description, Discovering, and Integration (UDDI) specification [168] has been used for this purpose in a solution involving Web Service as in [92]. The latter is to guarantee that all communication among all OADS components will be encrypted.

4.2.3. Alert Analyzer

Alert Analyzer is a tool that correlates incoming reports, trying to confirm the existence or not of attacks and anomalies. Moreover, it is also capable to predict future threats and targets. This OADS component has the function of receiving the aggregated alerts built by Alert Pre-Processor component, correlate them to increase their accuracy and consequently to validate the assumptions contained in each one of them, and possible predicting their occurrence in the future with some level of confidence.

The idea behind this correlation is to build an anomaly traffic patterns base. In other words, all confirmed positive diagnosis (true abnormal traffic) would generate

rules that can and must be stored, to be consulted on future. Although this may seem a bit controversial, there is a good reason for it. The following example presents to the reader a more detailed view. Assuming the existence of a network scenario where a host is infected by a botnet injecting a low-rate TCP SYN attack on the network. This attack occurs constantly at intervals of 10 minutes in a network where there are two detectors. The first detector (ADS1) was deployed to evaluate the TCP message exchange behavior pattern while the second one (ADS2) is specialized in massive traffic anomalies. When both are put together to evaluate this anomalous traffic, only ADS1 detects the abnormality because it recognizes a progressive growth in TCP connections as illustrated in Figure 4.4. As result, a generic rule will be created by OADS AC-Analyzer to identify this anomalous behavior clearly perceived on the picture.

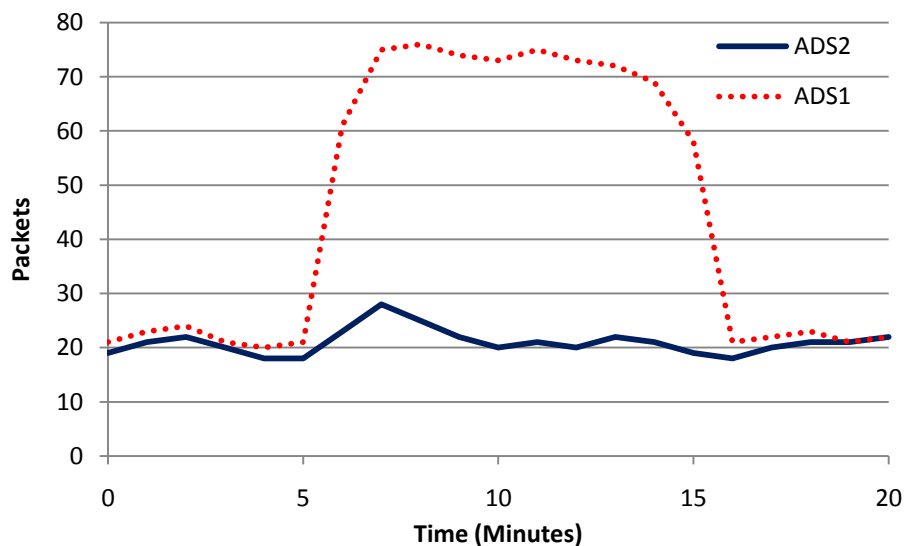


FIGURE 4.4: Example of attack evaluation by two anomaly detectors.

To achieve this objective, OADS Analyzer makes use of an adaptive technique, namely, episode frequency analysis [119][120]. It observes and develops knowledge, in the form of probabilistic rules, of the relationships among events that anticipate and make up a given attack. Not only is it capable of building adaptive event basis signatures, but can also be used to predict the buildup and preparation towards a possible attack before it is carried out, hence giving networks managers a kind of early warning system.

4.2.4. Decision Service

Decision Service (OADS-DS) implements the software responsible for the decision process related to analyzed network traffic. According to received information (or even collected), it decides whether there is any abnormality.

To implement the OADS-DS, a number of mechanisms, methodologies, and techniques can be used. Voting methods (simple voting, priority-base, weight levels, and decision tree), neural networks (supervised or unsupervised [169]), fuzzy logic [170], Dempster-Shafer's Theory of Evidence [171][172], Markov Chains (normal or hidden), finite state machines [173] and other knowledge fusion approaches are only some examples. The ability to tailor this service, making it context-aware, to different scenarios, network traffic, network status, available detectors, or other conditions is a great advantage of this service.

Since it has access and knowledge about the past and the current network status, OADS-DS uses predefined policies (heuristic rules) to choose the best decision mechanism to make at any determined moment. In OADS approach, Decision Service employs the concept of Policy-Based Management (PBM), defined by IETF [174], to take decisions according to defined network policies. Policy definitions are an answer to relatively high level questions such as: What to do when such event happens? The PBM approach is governed by a set of rules that determines the action course to be taken based on some conditions [175]. The evaluation of a policy is triggered by an event, which results in a policy decision being enforced on specific network device(s) or service(s). Policies are declarative, i.e., they can be adapted at run-time to flexibly control system behavior and are therefore becoming increasingly popular in adaptive, run-time configurable networks and information systems.

4.2.5. OADS Miner

OADS Miner is a specialized tool able to receive queries and to answer with summarized and specific content based on obtained information from the Internet and stored on anomaly and vulnerability base.

Basically, OADS Miner is divided in two distinct modules (Figure 4.5). The first one, named OADS Crawler, is responsible to gather on the Internet new information sources about traffic anomalies, vulnerabilities, and attacks. This module acts like a crawler collecting and concentrating the maximum possible information available on the Internet (technical and alert reports, summary traffic, black and white lists, and vulnerabilities databases) and stores them in a unique repository. It is important to emphasize that operation control like activation, deactivation, and parameters change (number of pages searched, initial URLs, specific content, for example) of OADS Crawler can and must be made by the Decision Service component. To achieve such goal, it is desirable that these actions are made using a predefined configuration file using a description language like XML.

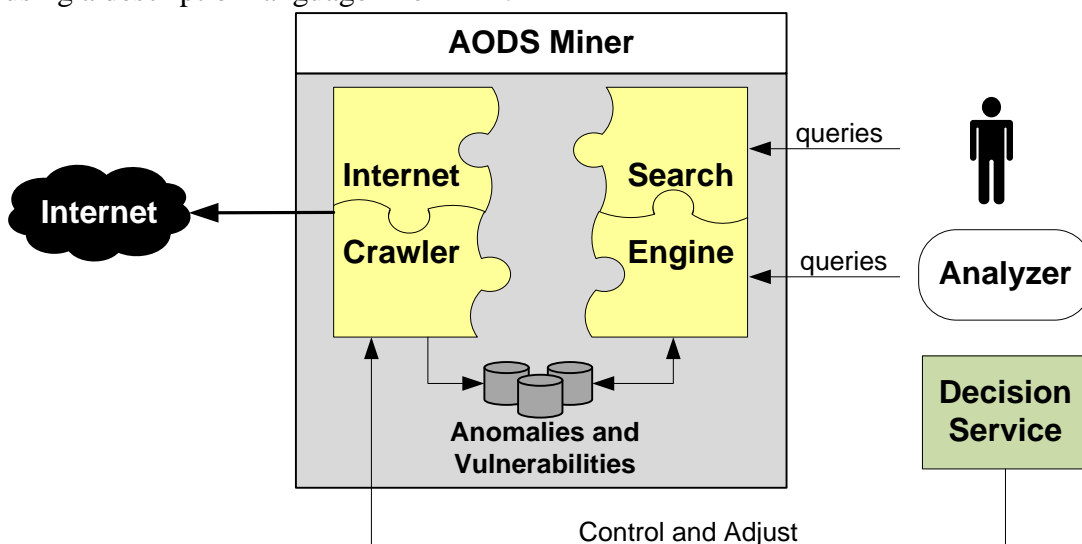


FIGURE 4.5: OADS Miner overview.

The second module works providing a differentiated search engine, focus on the support of the decision-making. It is capable to receive general or specific queries elaborated from end-users (typically network and IT managers) or other systems and tools, and return specific summarized information. This functionality is better explained

by the following example. Assuming that many anomaly detectors are reporting suspicious behavior in a determined set of IP address and that Alert Handler summarizes them in a same cluster, the Analyzer decides to search thoroughly what the possible causes are. For this, it consults the OADS Miner about information related with this case. So, OADS Miner executes queries in its anomaly and vulnerability base looking for any data which characteristics are similar or peculiar to the behavior. Hence, all available information matched with the original query is retrieved, summarized to eliminate duplications or inconsistencies, and forwarded to the Analyzer in order to decide whether the suspicious behavior is or not anomalous one.

4.3. OADS Contributions

The possibility of collaboration between anomaly detectors in the attempt to identify traffic anomalies quickly and accurately seems to be the major benefit of the OADS approach. Although, this modular framework may grow to embrace other detectors and technologies, it is seen as a good and significant step in the direction of self-defending networks.

Another benefit offered by the OADS approach is freeing the network operators and administrators of the routines and cumbersome repetitive tasks of evaluating security events, alerts, and incidents provided by the numerous network security software and services. Today, network security can no longer be achieved by individually setup of each network security element and the manual configuration of devices and services remains prone to human errors and does not scale well. For this reason, OADS approach may use security policies to especially describe or define what is considered as unwanted traffic and to support actions that deal with it (domain policies). In a recent work **Erro! Fonte de referência não encontrada.**, security policies were employed for this goal, although they are not very scalable.

The flexibility to update or add new detection techniques is another offered benefit. OADS approach permits that any detectors may participate into the framework since it makes use of a predefined pattern for communication exchanges (in this case IDMEF).

Lastly, OADS is a *generic and open approach* to deal with unwanted traffic. It is generic as its components may be assembled to detect a large number of different types of anomalies, spanning from a traditional LAN access control service to large high-speed ISPs and backbones. The approach is also open as it uses open source languages, defines and uses existing standard protocols, and tools. The possible number of anomaly detectors and their invocation sequence offers a wide and flexible range of effective cooperative work in order to defeat unwanted traffic. The external domain communication capability is no doubt a significant step in the direction of zeroing in on current cyber-attacks by rapid online cooperation with external security alert sites and colleagues.

4.4. Chapter Summary

Around two decades after the seminal work by Denning [176], anomaly detection remains a relatively less studied field when compared to intrusion detection for example. It is only now that the potential benefits of anomaly detection are being looked at under a high-speed network magnifying glass. This perspective culminates from the

increasing number of high-speed networks, the emergence of new services, and the increasing convergence of the two.

In this thesis, the collaboration and harmonization of different anomaly detectors is a step towards to achieve a desired effect in security area, i.e. turn the network more and more secure. In this context, this chapter presented a holistic view of the proposed architecture to deal with unwanted Internet traffic identification, called the Orchestration oriented Anomaly Detection System (OADS). This novel architecture employs collaboration and harmonization of different anomaly detectors to achieve a desired effect in the security area, i.e., turns the network more and more secure. In general lines, it facilitates the management of unwanted traffic identification, by providing means to integrate (collaboration) different anomaly detection techniques (detectors) and consequently increasing the network security level.

Although this thesis acknowledges that anomaly detection is still far from being solved, it believes that the most promising results will still be achieved and this thesis contributes with a new effective approach for anomaly detection. Moreover, as main benefit, it releases the network operators and administrators of the routines and cumbersome repetitive tasks of evaluating security events, alerts, and incidents provided by the numerous current network security software and sites.

Chapter 5

OADS Alert Pre-Processor

The growth of coordinated network attacks such as scans, worms and distributed denial-of-service (DDoS) is undoubtedly real. Although collaborative solutions have the potential to detect these attacks due to enabling all their sub-detection systems and sharing valuable intelligence with each other, some fundamental issues remain to be solved.

Sadoddin and Ghorbani [93] pointed out three reasons to justify the need for alert aggregation based solutions. First, it is not always easy to locate the source or target of attacks or faults in a network by examining merely low-level alerts. Secondly, current “low-level” data collection components consider raw alerts in isolation and raise alarms for each of them, without considering rich logical connections and relationship between these. Thirdly, automatic responses tend to be inefficient when solely based on such raw alerts as their input for decision taking.

The huge number of alerts triggered by numerous security and traffic analysis tools limits the ability to detect coordinated attacks in a scalable and accurate manner. Hence, the question is: how to aggregate and reduce duplicated alerts from different detectors inasmuch time as possible to permit their joint summarized interpretation?

This problem is addressed through the design and development of a component (tool) based on multi-source alert aggregation to deal with the generated huge volumes of raw alerts. Among the adopted techniques, there is one proposed by Xu *et al.* [151], focused as profiling Internet backbone traffic for discovering significant behavior patterns of interest. It provides their plausible interpretation, by aggregating raw alerts and extracting significant clusters along the three dimensions: source IP address, destination IP address and class of attack. An information-theoretic approach is taken to classify traffic into meaningful clusters.

The goal is to obtain the most relevant alerts grouped in clusters of interest, allowing that a correlation algorithm can be used to discover the attack strategy, helping network operators and IT managers to see the real attack intentions and take the most adequate decision. This solution has the potential to reduce the bandwidth and computational load at the (centralized) server, decreasing the false negative rate and prioritizing the most relevant alerts.

In order to explain how the proposed solution can be useful in the context of alert aggregation, the remainder of this chapter is structured as follows. First, the background behind entropy and relative uncertainty is explained, followed by the process of extracting significant clusters. Next, an overview design of the Alert Pre-Processor is presented, including complete details of each architectural component. Then, the implementation of each component is described. After, performance evaluation and stress tests are presented to validate the solution. Last, some conclusions are discussed.

5.1. Background

5.1.1. Entropy and Relative Uncertainty

The work of Xu *et al.* [151] proposed the use of an information-theoretical measure, named *relative uncertainty* (RU), to extract significant data based on the mathematical concept of *entropy*, proposed by Shannon and Weaver [152].

Entropy essentially measures “the amount of uncertainty” contained in determined information. Let X be a random variable that may take N_x discrete values $\{x_1, \dots, x_n\}$. Suppose we randomly sample or observe X for m times, which induces an empirical probability distribution on X , $p(x_i) = \frac{m_i}{m}$, $x_i \in X$, where m_i is the frequency or number of times that X was observed taking the value x_i [151]. The Shannon entropy of random variable X is defined as:

$$H(X) = - \sum_{i=1}^n p(x_i) \log p(x_i) \quad (1)$$

where by convention $0 \log 0 = 0$.

Since entropy measures the “observational variety” in the observed values of X , it is correct to affirm that $0 \leq H(X) \leq H_{max}(X) := \log \min(N_x, m)$, where $H_{max}(X)$ is defined as the maximum entropy of X when $p(x_i) = 1/n$.

So, assuming that there is an “observational variety” $m \geq 2$ and $N_x \geq 2$, Xu *et al.* [151] introduced a standardized entropy, named *relative uncertainty* (RU), to provide an index of variety or uniformity regardless of the support or sample size defines as:

$$RU(X) = \frac{H(X)}{H_{max}(X)} = H(X) / \log \min\{N_x, m\} \quad (2)$$

Since *relative uncertainty* provides an index of variety or uniformity in the observed values of X , if $RU(X) = 0$, then all observations of X are of the same kind, i.e., $p(x_i) = 1$ for some $x \in X$, meaning that the observational variety is completely absent. On the other hand, when $m \leq N_x$, $RU(X) = 1$ if and only if $|A| = m$ and $p(x_i) = 1/m$ for each $x_i \in A$, where A denotes a subset of the observed values of X . Thus all observed values of X are different or unique and the observations have the highest degree of variety or uncertainty. If $m > N_x$, $RU(X) = 1$ if and only if $m_i = m/N_x$, thus $p(x_i) = 1/N_x$ for $x_i \in A = X$, i.e., the observed values are uniformly distributed over X . In this case, $RU(X)$ measures the degree of uniformity in the observed values of X .

As described in their original work, the authors considered the conditional entropy $H(X|A)$ and conditional relative uncertainty $RU(X|A)$ by conditioning X based on A , where $H(X|A) = H(X)$, $H_{max}(X|A) = \log|A|$ and $RU(X|A) = H(X)/\log|A|$. Hence $RU(X|A) = 1$ if and only if $p(x_i) = 1/|A|$ for every $x_i \in A$. In general, $RU(X|A) \approx 1$ means that the observed values of X are closer to being uniformly distributed, thus less distinguishable from each other, whereas $RU(X|A) \ll 1$ indicates that the distribution is more skewed, with a few values more frequently observed. This measure of uniformity is used for defining “significant clusters of interest”

5.1.2. Extracting Significant Clusters

Before explaining the process used to extract significant clusters, it is necessary to clarify that the original focus has been changed. Instead of extracting clusters needed to identify attacks and anomalies, significant clusters are used to reduce the number of alerts and avoid the cognitive overloading of system managers.

In addition, for the above reason, unlike the original work that uses the four-feature space (source IP address, destination IP address, source Port, and destination Port) to determine the communication patterns of the end hosts and services, this work adopts a feature-space composed by the three elements, source IP address (*srcIP*), destination IP address (*dstIP*) and class of attack (*class*)²⁴. The extracted *srcIP* and *dstIP* clusters are used in a similar role to the one in the original work as they represent a set of “interesting” host behaviors (communication patterns), while the *class* cluster yields a set of “interesting” class/impact information of the attack alerts. No use is made of source Port and destination Port information in cluster identification due to the fact that source port can be easily changed to hide an attack and destination port is normally related with attack class field.

Regarding the process to extract “significant clusters of interest” proposed by Xu *et al.* [151], considering:

- X , a random variable. For example, representing a one feature dimension like *srcIP*,
- T , a time interval,
- m , the total number of alerts observed during the time interval T , and
- $A = \{a_1, \dots, a_n\}, n \geq 2$, the set of distinct values in X ,

the probability distribution P_A on X is given by $p_i = P_A = m_i/m$, where m_i is the number of alerts that take the value a_i and the (conditional) relative uncertainty, $RU(P_A) = RU(X|A)$, measures the degree of uniformity in the observed features A . If $RU(P_A)$ is close to 1, say, $> \beta = 0.9$, then the observed values are close to being uniformly distributed, and thus nearly indistinguishable. Otherwise, there are likely feature values in A that “stand out” from the rest.

Consequently, it is possible to define a subset S of A that contains the most significant (thus “interesting”) values of A if S is the smallest subset of A such that: i) the probability of any value in S is larger than those of the remaining values; and ii) the (conditional) probability distribution on the set of the remaining values, $R := A - S$, is close to being uniformly distributed. Intuitively, S contains the most significant feature values in A , while the remaining values are nearly indistinguishable from each other.

Algorithm 5.1 Simplified significant cluster extraction algorithm

Input: $\alpha := \alpha_0; \beta := 0.9; S := ;$

```

01:  $S := ; R := A; k := 1$ 
02: compute probability distribution  $P_R$  and its  $RU \theta = RU(P_R)$ ;
03: while  $\theta \leq \beta$  do
04:    $\alpha = \alpha \times 2^{-k}$ ;
05:   for each  $\alpha_i \in R$  do
06:     if  $P_A(\alpha_i) \geq \alpha$  then
07:        $S := S \cup \{\alpha_i\}; R := R - \{\alpha_i\}$ ;

```

²⁴ Class of attack is obtained from the IDMEF alerts.

```

08:     endif
09: end for
10:   compute (conditional) probability distribution  $P_R$  and  $\theta = RU(P_R)$ ;
11: end while

```

Algorithm 5.1 presents a general draft of the used algorithm (in pseudo-code) for extracting the significant clusters in S from A . Initializing with $\alpha_0 = 2\%$, the algorithm searches for the optimal cut-off threshold α^* from above via “exponential approximation” (reducing the threshold α by an exponentially decreasing factor $1/2^k$ with k constant). As long as the relative uncertainty of the (conditional) probability distribution P_R on the (remaining) feature set R is less than β , the algorithm examines each feature value in R and includes those whose probabilities exceed the threshold α into the set S of significant feature values. The algorithm stops when the probability distribution of the remaining feature values is close to being uniformly distributed ($> \beta = 0.9$).

The algorithm results are in the form of vectors, one for each cluster, containing the key of interest (IPsrc, for example), its frequency and the pointer for all elements this key.

It is important to explain that the Alert Pre-Processor component only employs the clustering technique from the proposed approach by Xu *et al.* [151].

5.2. Alert Pre-Processor Architecture

In order to achieve high detection accuracy without introducing an excessive computational overhead, an efficient mechanism of multi-source (multi-dimensional) alert aggregation is necessary.

The proposal addresses this issue through the design and implementation of an alert aggregation module (or component). This module receives alerts from different detectors, converts them to the IDMEF-based alert format, and then aggregates these using an adapted algorithm based on cluster similarity. Figure 5.1 depicts a functional diagram of the architectural components in accordance with their roles.

The main entities that compose Alert Pre-Processor are:

1. **Alert Handler Module** - plays a role that consists in receiving information (raw alerts) from different detectors and preparing these for analysis. It is seen as the front-end of the Alert Pre-Processor component since it represents the unique access gate to the offered functionalities. More specifically, it performs three activities:
 - *Translation* – performs translation of alert messages from different formats to the IDMEF standard format. This activity is optional.
 - *Validation* – performs the validation of the alerts to guarantee that they are in conformance with the IDMEF standard format.
 - *Ordering* – performs the ordering and synchronization of alerts according to their timestamps. It is important to emphasize the need of all detectors to have their clocks synchronized in order to ensure the correct ordering of alerts.

Architecturally speaking, the Alert Handler module is divided into two parts: Alert Handler agent (AHa) and Alert Handler server (AHs). The former runs continually when the detector is active, i.e., it is implemented as a daemon or

service. It is responsible for collecting the generated alerts, translate and validate them when necessary, and finally sending them to AHs. The second handler also acts as a daemon or service and is responsible for receiving the alerts from the AHa, ordering them according to their timestamps, and ultimately forwarding them to the Aggregation module.

2. **Aggregation Module** – is the core of the Alert Pre-Processor. It has as main task for aggregating alerts from multiples sources (detectors) that have some common feature values. It receives ordered alerts and executes a cluster-based algorithm in order to extract only significant alerts. The results (summarized alert information) are then forwarded for possible correlation to check or confirm the likelihood of attack or anomaly.

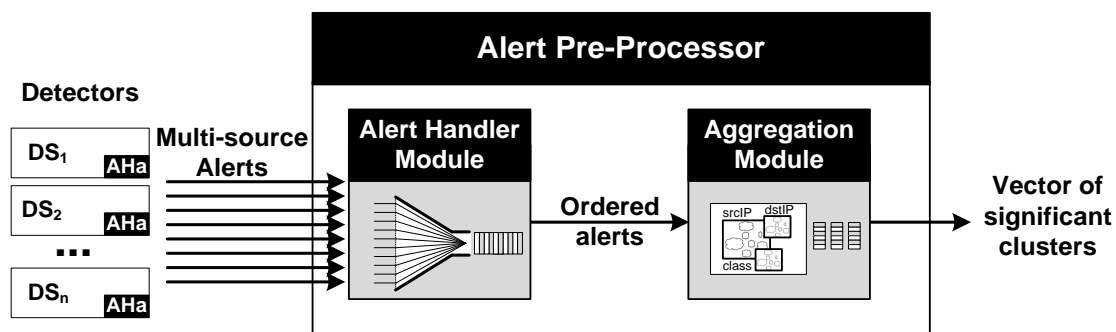


FIGURE 5.1: Functional diagram of OADS Alert Pre-Processor.

5.3. Implementation

This section describes the implementation process of the OADS Alert Pre-Processor while focusing on its data structure.

5.3.1. Alert Handler Module

In order to receive multi-source alerts, this handler employs a client/server approach, where the clients are called Alert Handler Agents (AHA) and the server is known as the Alert Handler Server (AHS).

Alert Handler Agent (AHA)

Developed using Java 1.6, AHAs are deployed together with anomaly and intrusion detectors (Snort, for example), in the form of a daemon processes. Their operation is divided in four tasks. The first one verifies the existence of alert files generated by detectors. To achieve this, an AHA checks periodically (typically between 1 and 5 minutes, usually configured according to the type of the detector used) if there are new alert logs. When it is the case, it copies these alerts and starts the processes of translation and validation.

The second task is that of translation. It converts the original alerts to the IDMEF format. For example, a typical output of the ChkModel [158] detector, composed by source IP address and port numbers, destination IP address and port numbers, the rate of sent and received packets and the state (see Figure 5.2a), needs to be translated for further manipulation. The result of the conversion to the IDMEF standard is shown in Figure 5.2b.



FIGURE 5.2: Translation example of ChkModel output to IDMEF format

In practice, AHAs have implemented translation support for three detectors: Prelude IDS (as previously explained in Chapter 4, section 4.3.2), Profiling [151] and ChkModel [158].

The third task is the validation of alerts. Its goal is to discard those alerts that are malformed. Basically, this task compares each alert with the IDMEF DTD²⁵ (Document Type Definition) to check whether or not it correctly formatted. The validation task is implemented using **DocumentBuilderFactory** class from the *javax.xml* package.

The fourth and last task consists of sending the alerts, via socket communication, to Alert Handler Server (AHS).

Alert Handler Server (AHS)

Also developed using Java 1.6 and designed to act as a service, AHS has the role of receiving, via socket communication, alerts from different detectors. All received alerts are buffered, ordered according to their timestamps, and then forward to the Aggregation module.

²⁵ DTD defines a structure of a document, where are specified what elements and attributes can be used in the document.

It is important to emphasize that this work considers that all detectors employ some kind of time synchronization like NTP (Network Time Protocol) to ensure the success of the overall approach.

5.3.2. Aggregation Module

The Aggregation module receives ordered alerts from the Alert Handler module and then processes them to extract the most significant ones. Since all received alerts are considered “important”, two data structures using Java, named *ATable* and *CTable*, were implemented to store some values of interest for further evaluation. Figure 5.3 illustrates the structures of *ATable* and *CTables*.

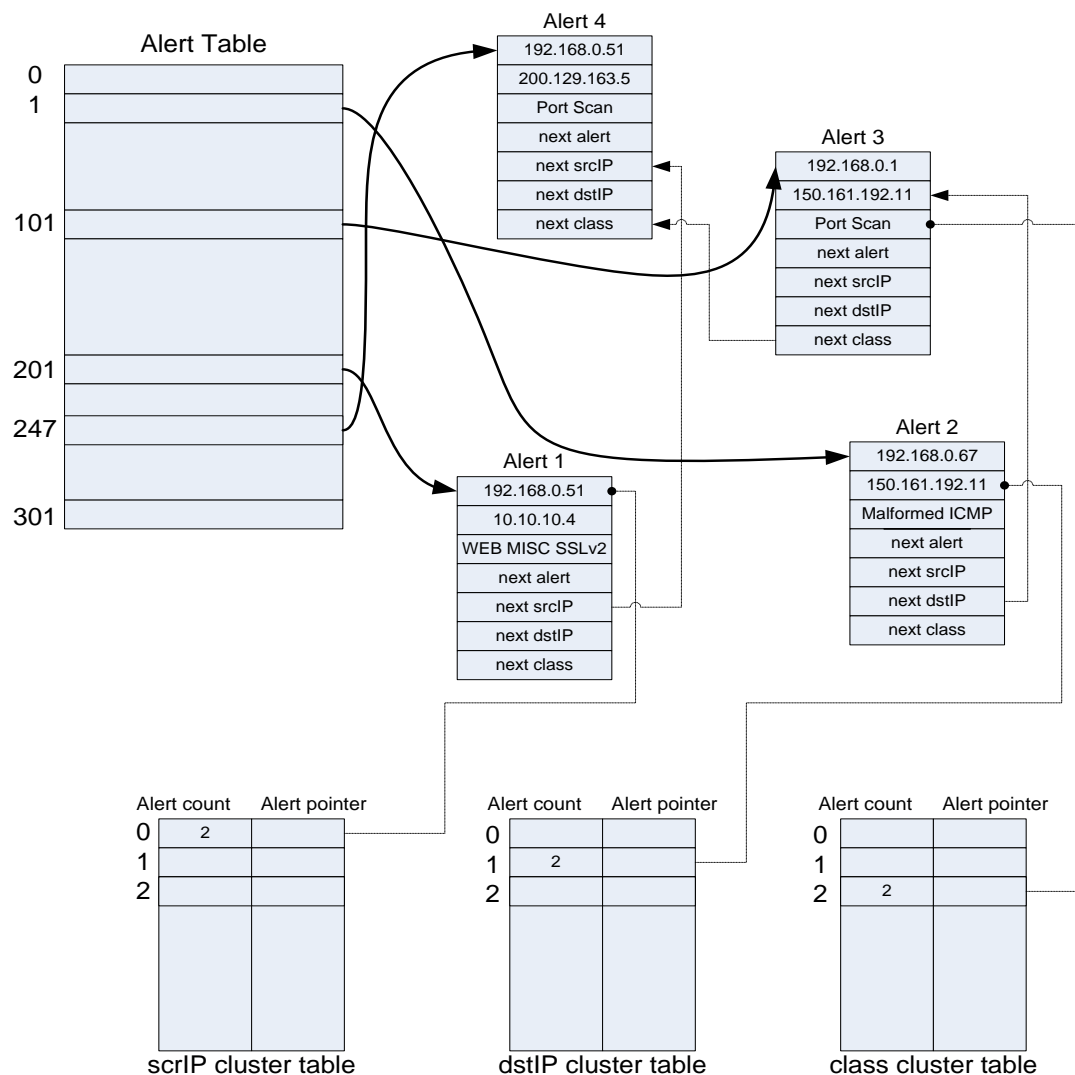


FIGURE 5.3: Data structure of *ATable* and *CTable*

ATable is an array data structure that stores source IP address information (*srcIP*), source port, destination IP address (*dstIP*), destination port, class of attack (*class*), timestamp and alert severity. In addition, it also has three alert pointers (*next srcIP*, *next dstIP*, and *next class*) to link alerts sharing the same feature value in the given dimension. This idea removes the need to duplicate the alerts and then to group each alert into three clusters along each dimension, not to mention that is both more scalable and efficient regarding memory cost especially when dealing with hundreds of millions of alerts. For example, in Figure 5.3, the *next srcIP* pointer of *Alert 1* links to

Alert 4 since they share the same source IP *192.168.0.51*. Similarly, the *next dstIP* pointer of *Alert 2* links to *Alert 3* since they share the same destination IP *150.161.192.11*, and the *next class* pointer of *Alert 3* links to *Alert 4* since they share the same class *Portscan*.

Once all the received alerts are correctly stored in *A*Table, *C*Tables are put in operation to reference the first occurrence of an alert, providing an easy and simple way to quickly find the “old” alerts of the same clusters. Since there are three types of clusters, three instances of *C*Table were created for managing clusters along three dimensions.

In spite of their simple design, *C*Tables are essential to the computation and extraction of the significant clusters. Each *C*Table stores an alert counter, for recording the number of occurrences of a given value, and an alert pointer, for referencing the its first time. For example, when evaluating *alert 1* in Figure 5.3, the given source IP address (*150.161.192.51*) is compared with *srcIP* cluster and as it is the first time that this IP address occur it is therefore inserted into *C*Table. The alert count field is incremented by 1 and an alert pointer is linked to this alert at *A*Table. The same occurs with the other dimensions *dstIP* and *class*. However, when evaluating *alert 3*'s *dstIP* value (*150.161.192.11*), one finds the first alert (*alert 2*) of the cluster *dstIP* (index 1), and updates the *next dstIP* pointer of *alert 2* to *alert 3*. Next, alert count is finally incremented by 1.

When the *C*Tables are filled (with the insertion of all *A*Table elements), the process of cluster extraction is triggered. According to algorithm 5.1, it results in three lists (one for each cluster) composed by key (*srcIP*, *dstIP* or *class of attack*), frequency and the pointer for the first occurrence in its respective *C*Table.

As a final result of Aggregation module, each list is used to create a vector (one for each dimension) containing only significant elements (alerts). Each vector is composed by the following attributes: source IP address, source port, destination IP address, destination port, class of attack, timestamp and alert severity.

5.4. Performance Evaluation

5.4.1. Benchmarking

CPU load and memory usage of the Aggregation module are measured using jProfiler [177], an all-purpose java profiling suite targeted at Java applications. Its features include CPU, memory and thread profiling telemetry.

In order to collect the CPU load, the code has been divided in two parts: *A*Table construction and significant cluster extraction, which also includes *C*Table construction. Similarly, the same division is applied to measure memory usage. Next, a number of breakpoints before and after each part are inserted. The output difference points out the CPU load and memory consumption of each part whereas the sum of these parts indicates the total CPU load and memory usage of the Aggregation module.

To test the Aggregation module two alert files (*AF1* and *AF2*) inject anomalous traffic (port scans and traffic with the same source and destination address) in the used test-bed consisting of real machines within the GPRT research laboratory. Table 5.1 summarizes the characteristics of these alert files.

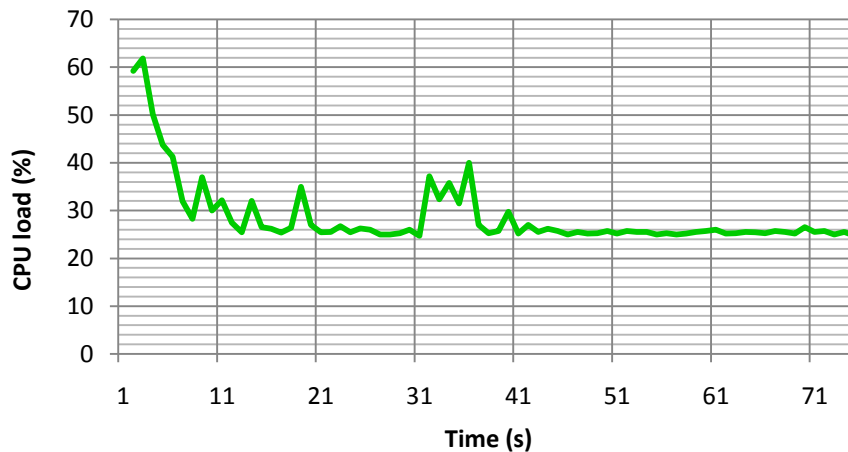
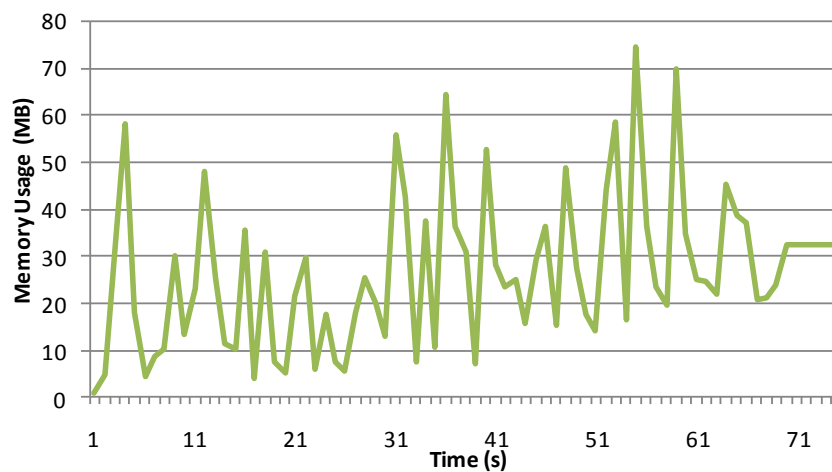
TABLE 5.1: Characteristic of alert files AF1 and AF2

File	Date	Duration	No. Alerts	Size	Detector
AF1	November, 30 2009	01h 45min	5.224	12.6 Mb	Snort 2.8.3
AF2	December, 1 2009	06 h 04 min	24.824	36.5 Mb	Snort 2.8.3

Table 5.2 shows CPU load and memory usage results for these two alert files (AF1 and AF2). Figures 5.4 and 5.5 show the CPU load and memory cost of the Aggregation module for AF2 (the largest alert file), respectively. Interestingly, it is possible to clearly observe that the greater the number of alerts to be processed the smaller is the CPU load and memory usage of the module. Such fact could be due to the reading of alerts fields to build the *ATable*.

TABLE 5.2: CPU load and memory usage of the Aggregation module

File	CPU Load (%)			Memory (MB)		
	min	avg	max	min	avg	max
AF1	22,47	34,51	64,43	1,4	29,13	84,72
AF2	24,75	28,33	61,81	0,83	26,62	74,37

**FIGURE 5.4: CPU load of the Aggregation module from alert file 2 (AF2)****FIGURE 5.5: Memory usage of the Aggregation module from alert file 2 (AF2)**

In summary, the CPU load tends to be constant and is largely determined by the number of alerts to record in *A*Table and consequently in *C*Tables as well as that of the size of significant clusters. On the other hand, memory usage is determined by the number of alerts to be evaluated and recorded in *A*Table, as represented by the time interval between 1 and 69 seconds in Figure 5.5. The oscillation perceived is a consequence of the process of the alerts reading and evaluation process and their insertion in *A*Table. The time interval between 69 and 75 in both pictures correspond to *C*Tables building, significant cluster extraction and the generation of the resulting vectors.

5.5. Stress Test

The performed performance benchmarking of CPU load and memory usage demonstrated the operational feasibility of the aggregation module. However, one needs to show that it is efficient when extracting significant clusters. In order to fulfill this requirement, two experiments were conducted: the first using the alert files of performance benchmarking and the second using a DARPA 2000 dataset [179]. The latter is a known publically available trace of security attacks and has been used in many works [103][130][132]. It is important to emphasize that in both tests, the IDS Snort (version 2.8.3.2) [163] is running to capture anomalies and the Snort-IDMEF plugin [178] is used to translate Snort logs into IDMEF alerts.

5.5.1. Alert files AF1 and AF2

Table 5.3 summarizes the results obtained by the implemented Aggregation module from AF1 and AF2. Note that both use the IDMEF format.

TABLE 5.3: Characteristic of alert files AF1 and AF2

File	Src IP		Dst IP		Class	
	Unique	Extracted	Unique	Extracted	Unique	Extracted
<i>AF1</i>	2	1	467	13	8	4
<i>AF2</i>	35	20	22	7	16	7

From alert file 1 (AF1), consisting of 1399 alerts, 4 significant clusters of attack classes from a total of 8, 13 significant clusters of destination IP addresses from a total of 467 and 1 significant cluster of source IP address from a total of 2 distinct IP addresses have been extracted. When processing alert file 2 (AF2), made of 24.824 alerts, 7 significant clusters of class of attack from a total of 16, 7 significant clusters of destination IP address from a total of 22 and 20 significant clusters of source IP address from a total of 35 distinct elements were also extracted. Figure 5.6 and Table 5.4 illustrate the class of significant attack clusters extracted from AF2.

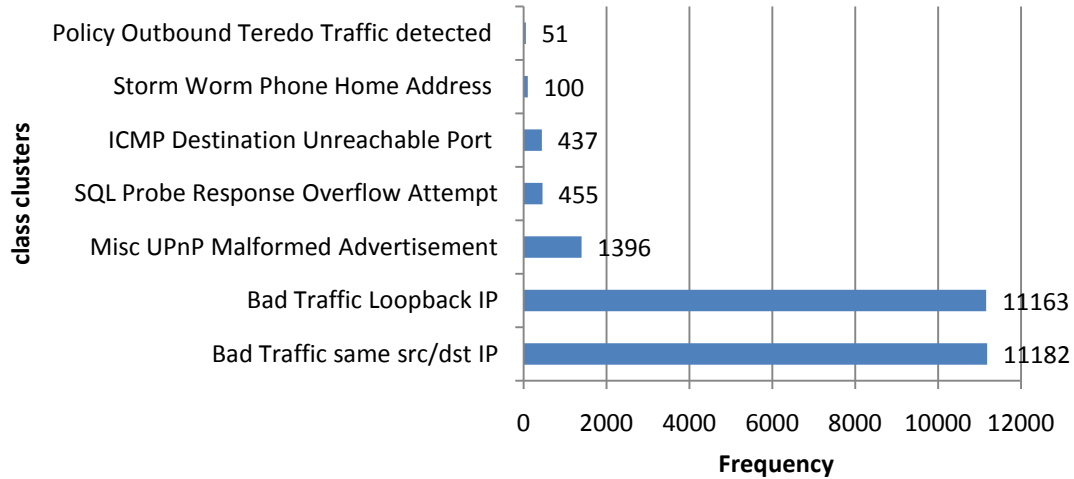


FIGURE 5.6: Significant clusters extracted from class of attack dimension in AF2

TABLE 5.4: Class of Attack relative uncertainty in AF2

Class of Attack	RU	Probability [$P_A(\alpha_i)$]	α
<i>Bad Traffic same src/dst IP</i>	0.387171712517231	0.450451176281018	0.02
<i>Bad Traffic Loopback IP</i>	0.396398827192577	0.449685787947147	0.02
<i>Misc UPnP Malformed Advertisement</i>	0.406761881088819	0.056235900741218	0.02
<i>SQL Probe Response Overflow Attempt</i>	0.418514276148485	0.018329036416371	0.01
<i>ICMP Destination Unreachable Port</i>	0.519671597203675	0.017603931679020	0.01
<i>Storm Worm Phone Home Address</i>	0.602436402138431	0.004028359651950	0.0025
<i>Policy Outbound Teredo Traffic detected</i>	0.627372862460001	0.002054463422494	0.00125
Final RU	0.925323538573275		

Table 5.4 represents the extraction of significant clusters based on relative uncertainty (RU) according to the previously described Algorithm 5.1. Before initiating the extraction process, the first RU is calculated while considering all elements inside the set, where $RU(A) = 0.387171712517231$. When the first cluster (*bad traffic loopback ip*) begins to be evaluated, its probability is calculated by dividing their frequency by the total number of alerts (522/1109), resulting in $P_A(\alpha_1) = 0.450451176281018$. After that, its probability is compared with the α parameter (line 6 of the Algorithm 5.1). If its probability is greater than or equal to α , then this cluster is significant and is therefore added to the set S and removed from set R . This process is repeated until all elements of R have been compared to α . In Table 5.4, the three first clusters were extracted in the first interaction ($\alpha = 0.02$), whereas the fourth, fifth, sixth and seventh clusters are extracted in the second ($\alpha = 0.01$), third ($\alpha = 0.0025$) and fourth ($\alpha = 0.00125$) iterations respectively. Note that in each interaction, α is reduced by half, to improve the approximation. The extraction of significant clusters stops when the RU value is greater than β , in this case $0.925323538573275 > 0.9$.

Figures 5.7 and 5.8 illustrate the destination and source IP address significant clusters extracted from AF2.

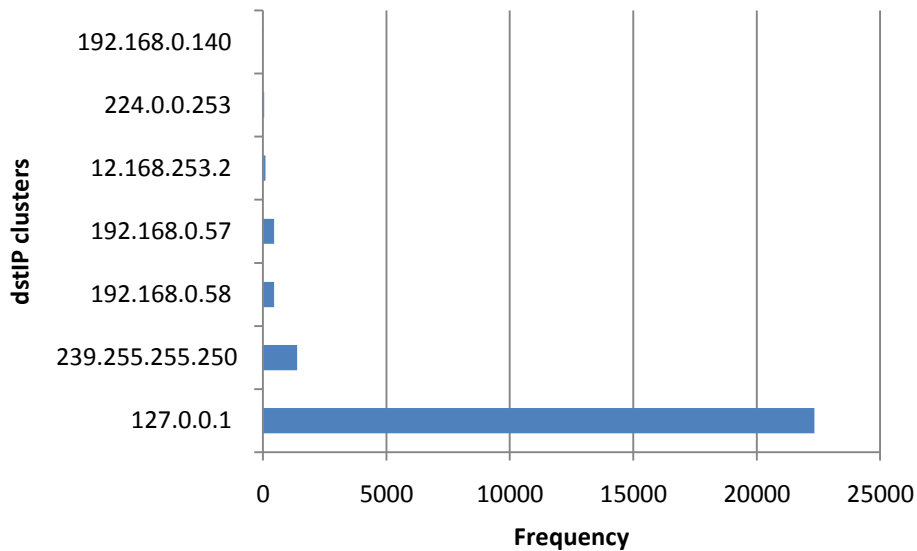


FIGURE 5.7: Significant clusters extracted from destination IP address dimension in AF2

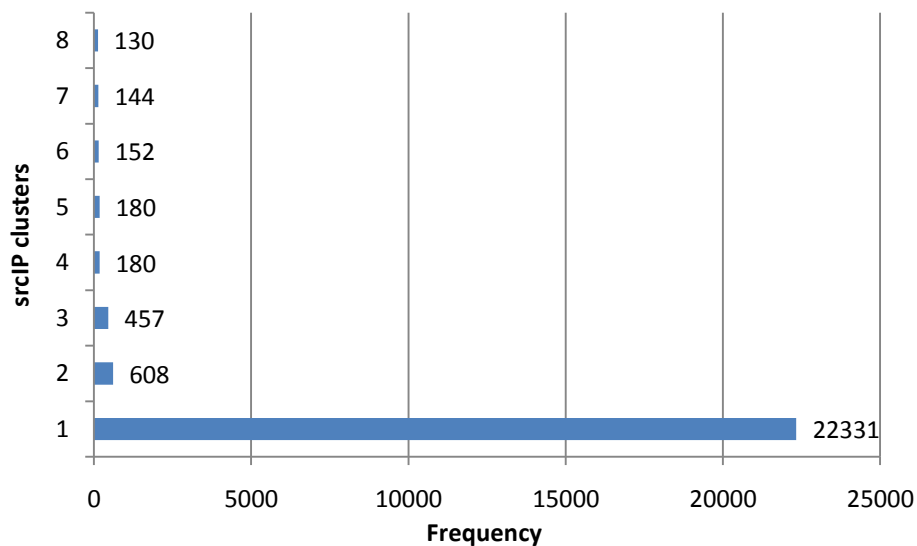


FIGURE 5.8: Significant clusters extracted from source IP address dimension in AF2

5.5.2. DARPA 2000 Dataset

The DARPA 2000 dataset [179] is a known trace offered as an IDS baseline evaluation. Created by the MIT Lincoln Laboratory, it contains two scenarios (LLDOS 1.0 and LLDOS 2.0.2), where both present traffic from external and internal networks.

LLDOS 1.0 is divided in 5 phases. In *phase 1*, the attacker sends ICMP to discover which hosts are active in the targeted network. The packets are sent to sub-networks 172.16.115.0/24, 172.16.114.0/24, 172.16.113.0/24 and 172.16.112.0/24. Once with a list of active hosts as a result of the previous scanning phase, the attacker begins *phase 2* launching an exploit type tool to determine if the *sadmind* service is executing at one of these hosts in the active list. *Phase 3* consists in a set of attempts to gain privileged root mostly using known buffer-overflow attacks over hosts executing the *sadmind* daemon service. If root access to a host is obtained, the attacker initiates *phase 4* using remote access commands such as telnet and remote procedure calls (RPC) to launch DDoS attacks from the newly invaded hosts. A new file named “.rhosts” (short or remote hosts) and a program called *master-sol* are installed into these hosts. To

finalize (*phase 5*), the attacker, controlling the three hosts (172.16.115.20, 172.16.112.10 and 172.16.112.50), uses the command “*mstream 131.84.1.31 5*” to unleash large numbers of packets to the target 131.84.1.31 during 5 seconds while using random IP source addresses to avoid detection in what is known as IP spoofing.

Similarly to LLDOS 1.0, the **LLDOS 2.0.2** DDoS dataset consists of 5 phases. In *phase 1*, the attacker made HINFO DNS queries from the DNS server (172.16.1145.20) trying to obtain information about possible future victims. The knowledge of the hardware and software environment of a possible victim allows the attacker to better narrow down the techniques and tools that are more likely to succeed. In *phase 2*, using the same *sadmind* vulnerability, the attacker invades the DNS server. Next, *phase 3*, through an FTP connection, the attacker installs remotely the *mstream* program in the DNS server. The attacker also tries to gain control of two other hosts, but in only one it succeeds (*phase 4*), where *mstream* program is installed. Lastly (*phase 5*), using both controlled hosts, DDoS packets are sent towards the same host LLDOS1.0 (131.84.1.310) for the duration of 5 seconds while using fake random source addresses.

Results

In LLDOS 1.0 scenario, 1109 internal alerts (*inside-tcpdump* file) and 2,465 external alerts (*dmz-tcpdump* file) were evaluated. It is important to emphasize that Snort was not able to detect phase 1 and part of phase 4 of this scenario. The explanation is simple. Snort does not consider (and has no rules for) ICMP requests and telnet connections as malicious activities. All other phases are detected.

Specifically, using internal alerts, from a total of 29 distinct classes of attack (*class*), 37 distinct destination IP address (*dstIP*) and 294 distinct source IP address (*srcIP*), the Aggregation module extracted 9 significant clusters of class of attacks, 12 significant clusters of destination IP address and no significant cluster of source IP address. Figures 5.9 and 5.10 illustrate the frequency distribution of the extracted clusters of *class* of attack and destination IP address respectively, and Tables 5.5 and 5.6 summarize the relative uncertainty (RU) for these two dimensions (*class* and *dstIP*).

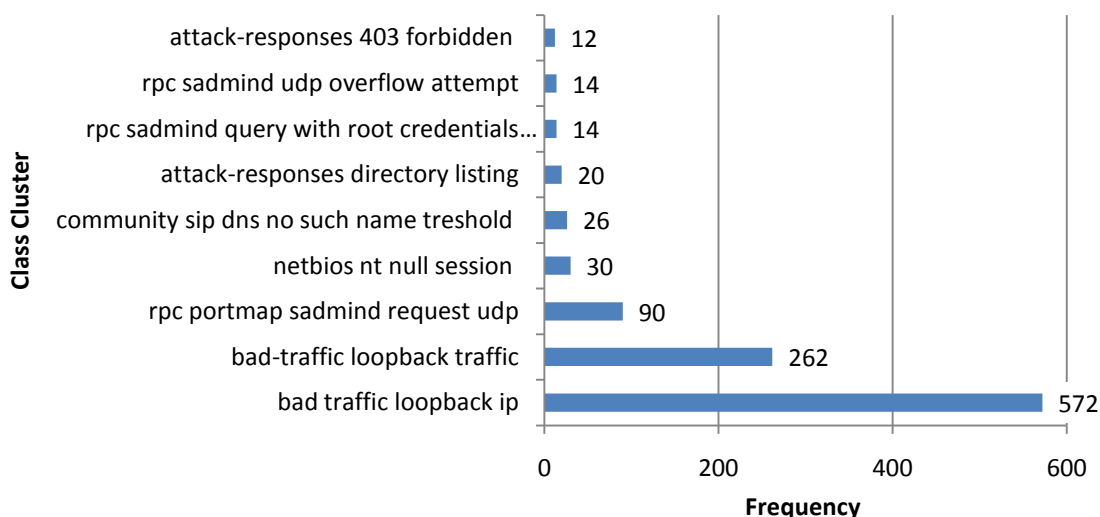


FIGURE 5.9: Class of attack frequency distribution in LLDOS 1.0 inside scenario

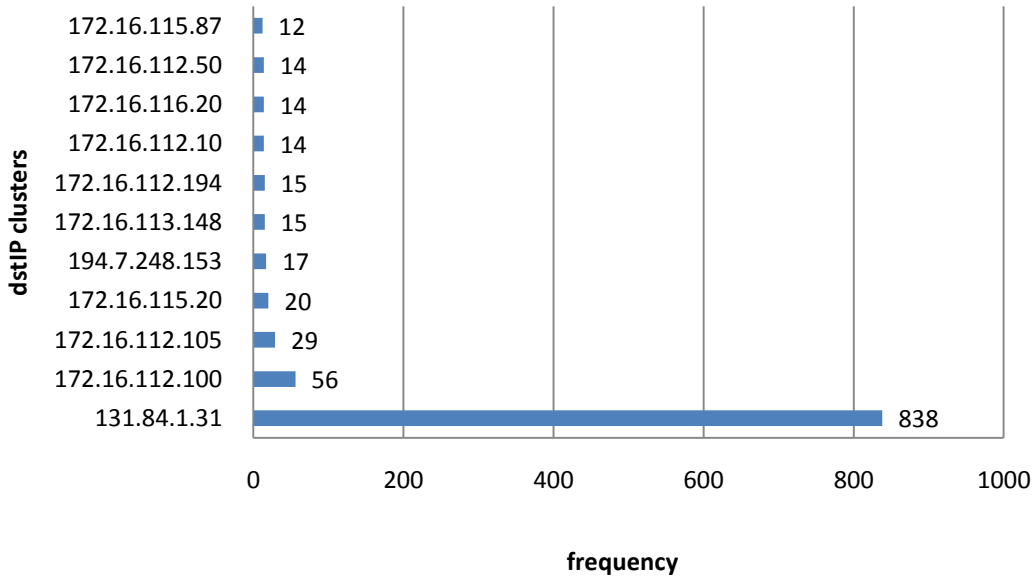


FIGURE 5.10: Destination IP frequency distribution in LLDOS 1.0 inside scenario

TABLE 5.5: Class of attack relative uncertainty in LLDOS 1.0 inside scenario

Class of Attack	RU(class)	Probability [$P_A(\alpha_i)$]	α
<i>bad traffic loopback ip</i>	0.488631524175470	0.5157799819657349	0.02
<i>bad-traffic loopback traffic</i>	0.493777284296841	0.23624887285843102	0.02
<i>rpc portmap sadmind request udp</i>	0.499225829629953	0.0811541929666366	0.02
<i>netbios nt null session</i>	0.505008637575596	0.027051397655545536	0.02
<i>community sip dns no such name threshold</i>	0.511161965629217	0.023444544634806132	0.02
<i>attack-responses directory listing</i>	0.517727823878885	0.018034265103697024	0.01
<i>rpc sadmind query with root credentials attempt udp</i>	0.886411053116980	0.012623985572587917	0.01
<i>rpc sadmind udp overflow attempt</i>	0.899158381707709	0.012623985572587917	0.01
<i>attack-responses 403 forbidden</i>	0.912897436954964	0.010820559062218215	0.01
Final RU	0.9277653930176893		

TABLE 5.6: Destination IP relative uncertainty in LLDOS 1.0 inside scenario

Destination IP	RU (dstIP)	Probability [$P_A(\alpha_i)$]	α
131.84.1.31	0.349490430592109	0.7556357078449053	0.02
172.16.112.100	0.352162574775310	0.05049594229035167	0.02
172.16.112.105	0.354952942524035	0.026149684400360685	0.02
172.16.115.20	0.357870741608349	0.018034265103697024	0.01
194.7.248.153	0.860696355068559	0.015329125338142471	0.01
172.16.113.148	0.868338326412219	0.013525698827772768	0.01
172.16.112.194	0.876366490152678	0.013525698827772768	0.01
172.16.112.10	0.884815250608347	0.012623985572587917	0.01
172.16.116.20	0.893723469979750	0.012623985572587917	0.01
172.16.112.50	0.903135238078656	0.012623985572587917	0.01
172.16.115.87	0.913100810499855	0.010820559062218215	0.01
172.16.113.105	0.923677760466658	0.010820559062218215	0.01

Final RU	0.934932404156025		
-----------------	--------------------------	--	--

Regarding source IP address information, no significant cluster could be extracted since the initial RU of the set is 0.922710283397728, that is, $RU(A) > 0.9$. This happens because although there are 294 clusters their frequency distributions have low variation. Figure 5.11 shows the dispersion of source IP address clusters.

For external alerts observed in the LLDOS1.0 scenario, from a total of 24 distinct classes of attack (*class*), 42 distinct destination IP address (*dstIP*) and 29 distinct source IP address (*srcIP*), the Aggregation module extracted 9 significant cluster of *class* of attack, 10 significant clusters of *destination IP address* and 4 significant cluster of *source IP address*. Figures 5.12, 5.13 and 5.14 illustrate the frequency distribution of the extracted clusters

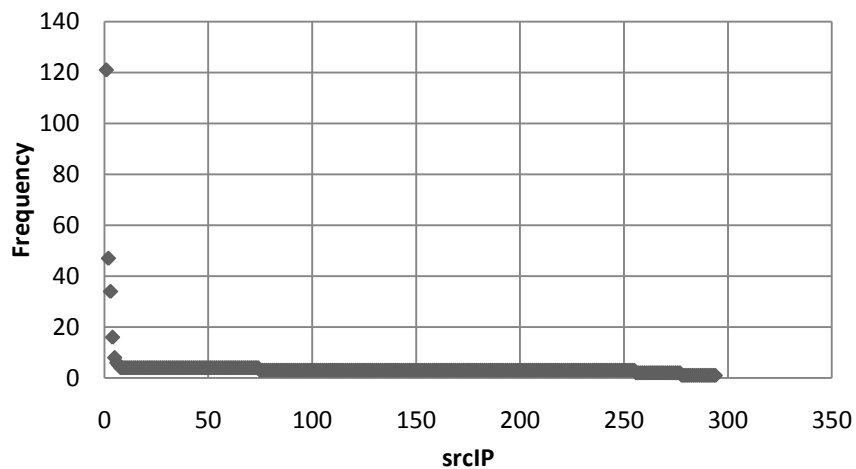


FIGURE 5.11: Source IP clusters dispersion in LLDOS 1.0 inside scenario

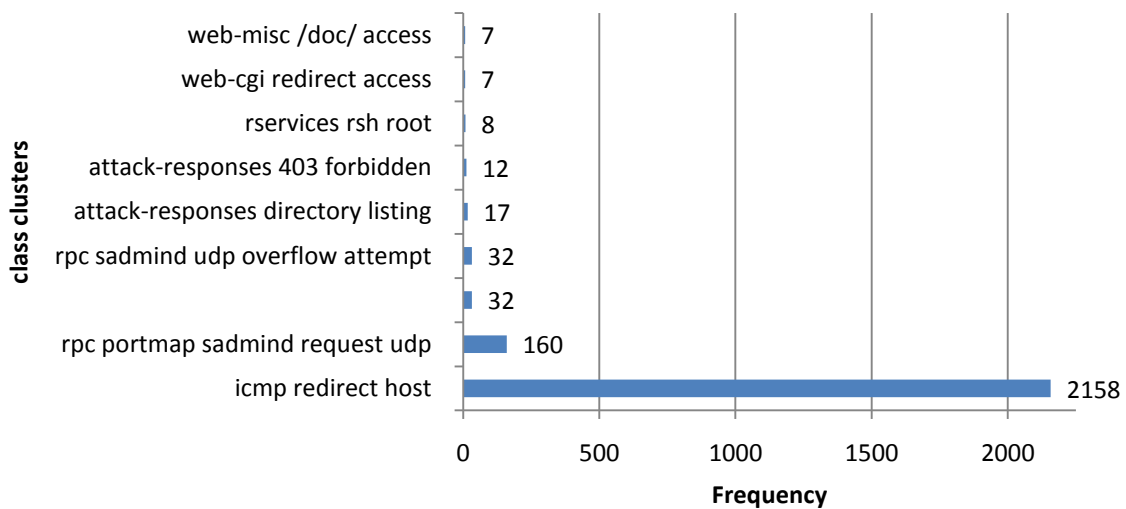


FIGURE 5.12: Class of attack frequency distribution in LLDOS 1.0 outside scenario

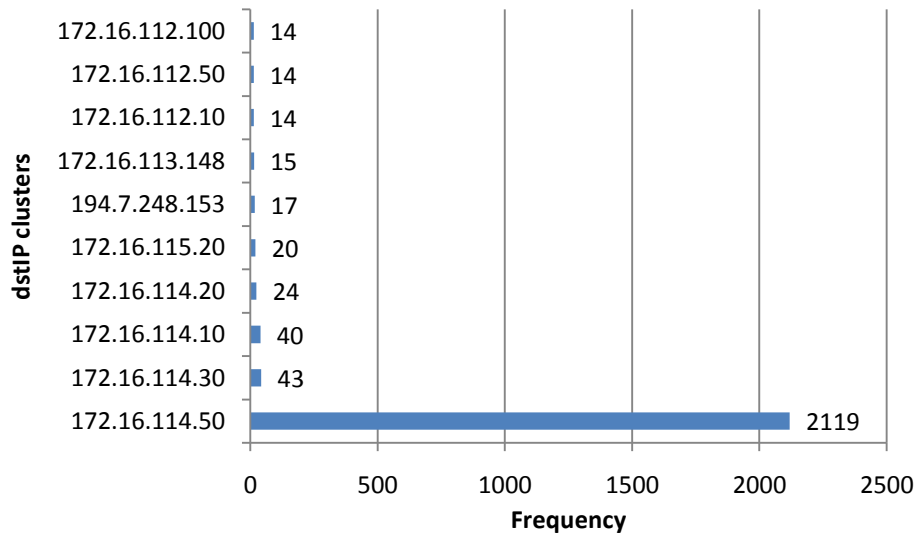


FIGURE 5.13: Destination IP frequency distribution in LLDOS 1.0. outside scenario

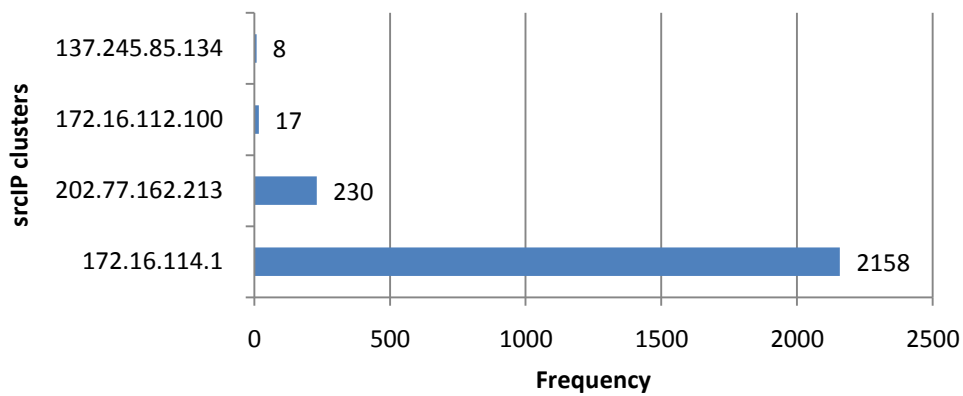


FIGURE 5.14: Source IP frequency distribution in LLDOS 1.0 outside scenario

In the LLDOS 2.0.2 scenario, 935 internal and 1.108 external alerts were evaluated. Similarly to the first scenario, phase 1 was not detected, whereas phase 3 did not generate any alert. All other phases are detected. Using internal alerts, from a total of 29 distinct classes of attack (*class*), 26 distinct destinations IP address (*dstIP*) and 434 distinct sources IP address (*srcIP*), the Aggregation module extracted 6 significant cluster of *class* type attack, 2 significant clusters of *destination IP address*. No significant cluster of *source IP address* was detected as the initial RU of the set is 0.9803433665539428, that is, $> \beta = 0.9$. Figures 5.15 and 5.16 illustrate the frequency distribution of the extracted clusters of *class* and *destination IP address* types of attacks respectively.

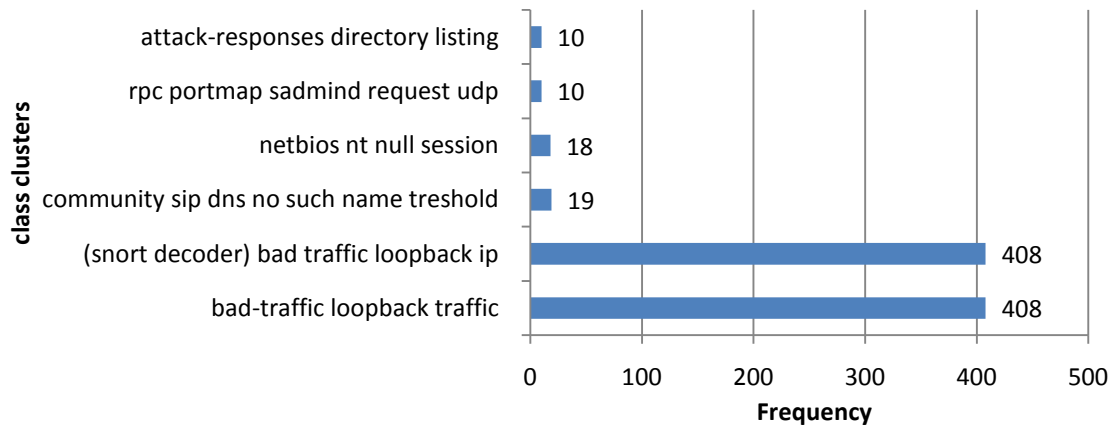


FIGURE 5.15: Class of attack frequency distribution in LLDOS 2.0.2 inside scenario

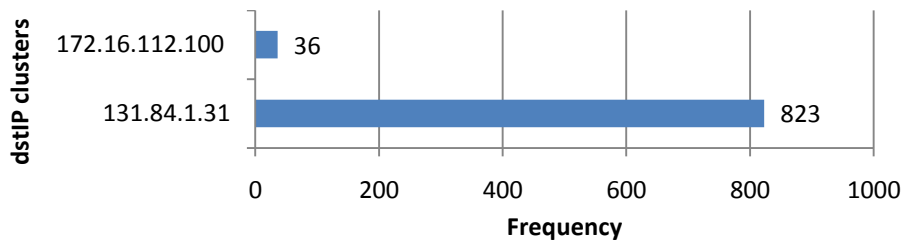


FIGURE 5.16: Destination IP frequency distribution in LLDOS 2.0.2 inside scenario

For external alerts in the LLDOS2.0.2 scenario, from a total of 23 distinct classes of attack (*of type class*), 28 distinct destination IP address (*dstIP*) and 20 distinct source IP address (*srcIP*), the Aggregation module extracted 5 significant clusters of *class type* attacks, 8 significant clusters of *destination IP address type* and 1 significant cluster of *source IP address* (IP address 172.16.114.1 had as many as 1047 occurrences and $RU=0.120516348873754$). Figures 5.17 and 5.18 illustrate the frequency distribution of the extracted clusters

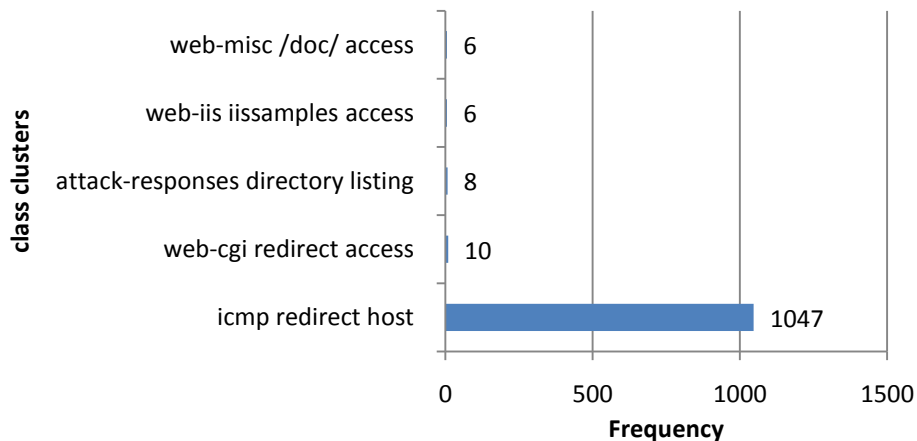


FIGURE 5.17: Class of attack frequency distribution in LLDOS 2.0.2 outside scenario

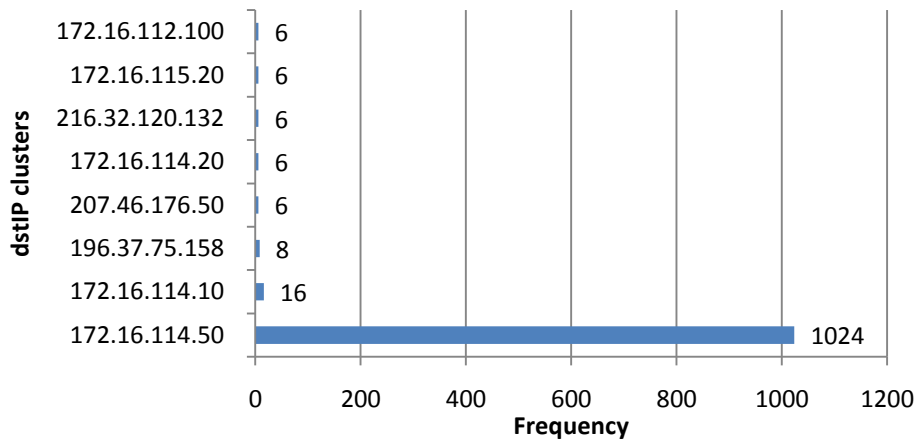


FIGURE 5.18: Destination IP frequency distribution in LLDOS 2.0.2 outside scenario

5.6. Chapter Summary

As discussed previously, collaborative solutions are characterized by a huge amount of generated alerts. To deal with this issue, the availability of efficient storage and search schema is necessary.

This chapter discussed the design and implementation of a solution, named Alert Pre-Processor, capable of convert original alerts into uniform IDMEF-based alerts and summarizing or aggregating these using a clustering algorithm. It therefore allows a more efficient alert correlation and consequently improves the potential usage within a collaborative framework. The cluster-based approach proposed by Xu *et al.* [151] to extract clusters of interest from raw intrusion alerts has been adopted.

To evaluate the proposed solution, some real-world intrusion data sets were collected from the GPRT research laboratory in addition to the DARPA 2000 dataset. As demonstrated, the proposed solution reduces alert messages significantly.

Chapter 6

OADS FER Analyzer

Although there are many available solutions for detecting anomalous traffic, including IDS (Intrusion Detection Systems), IPS (Intrusion Prevention Systems) and APS (Anomaly Prevention Systems), their effectiveness depends mostly on a higher level of management and coordinated usage.

Currently, different efforts have been made to develop collaborative solutions, called CIDSs (Collaborative Intrusion Detection Systems) and CAIDSs (Collaborative Anomaly and Intrusion Detection Systems) composed by a set of individual IDSs, IPSs and APSs coming from different network administrative domains or organizations, which cooperate to detect coordinated attacks. A key component of the proposed solution in this work is the alert correlation mechanism it uses. This clusters similar incidents observed by different IDSs, prioritizes these incidents, and identifies false alerts generated by individual IDSs. As result, a global and condensed high-level view of network attacks resulting from analyzing raw alerts is then produced.

However, one of the issues involving alert correlation work (also known as event correlation) lies in the need to improve the scalability of alert correlation while still maintaining the expressiveness of the patterns that can be found. According to [180], solutions based on *single-dimensional* correlation have been widely used due to their simplicity, but they fail to characterize a wide scope of types of attack behaviors. Multi-dimensional correlation schemas that are capable of identifying more patterns in events provide better solutions. Although *multi-dimensional* correlation has a clear advantage in terms of their expressiveness, its computational complexity limits its use in collaborative IDSs, especially when operating online.

In this thesis, the alert correlation problems were addressed through the design and development of a system based on data mining. The present solution uses the concept of Frequent Episodes Rules (FER) to perform sequence analysis and consequently detect anomalies, including also unknown attack patterns. Proposed originally by Mannila *et al.* [120] for monitoring alarms in telecommunication networks and finding relationships among them, FER is based on the fact that the data subject to analysis consists of a sequence of events. So, the question is to find into collections of events those that occur frequently together. FER is employed to observe and develop a specific knowledge, in the form of probabilistic rules, of the relationships among events (alerts) that anticipate and make up a given attack or anomaly. Moreover, it is capable of building adaptive event basis signatures, but it also can be used to predict the buildup and preparation towards a possible attack before it is actually carried out, hence giving networks managers a kind of early warning system.

In order to explain how the following solution can be useful in anomaly and attack detection and their early intercept, the remainder of this chapter is structured as follows. Firstly, the theory behind frequent episodes analysis is introduced and some examples of its applicability are discussed. Next, an overview design of the proposed solution is presented, including a complete detail of each architectural component and

its accomplished implementation. Then, an initial evaluation is presented to validate the solution. Lastly, some conclusions are discussed to summarize this chapter results.

6.1. Frequent Episodes Rules (FER)

Mannila *et al.* [120] proposed and designed a popular framework for spatial and temporal data mining, named frequent episodes rules. The framework is applicable on data as a single long sequence of ordered pairs, which are called events. Two key concepts are adopted in frequent episodes analysis: event sequence and episode. The first one refers to user/system actions and behavior collected over many domains or places when the second one is seen as a set of events occurring relatively within small distances following some partial order.

Next, the basic concepts of frequent episodes discovery employed in anomaly detection and their prediction, including event sequence, time window, episodes and sub-episodes, and calculations of frequency are presented.

6.1.1. Basic Concepts

Event Sequence

According to Mannila *et al.* [120], given the set E of event (A, t) , where $A \in E$ is its type (which takes values from a finite alphabet, A) and t an integer representing its occurrence instant.

An event sequence s over E is the set (s, T_s, T_e) , where

$$s = \langle (A_1, T_1), (A_2, T_2), \dots, (A_n, T_n) \rangle$$

is an ordered event sequence such that $A_i \in E$ for all $i = 1, \dots, n$, and $t_i \leq t_{i+1}$ for all $i = 1, \dots, n - 1$. Furthermore, T_s and T_e are two integers that represent the starting and terminating times, respectively, and $T_s \leq t_i < T_e$ for all $i = 1, \dots, n$.

Figure 6.1 illustrates the sequence of events $s = (s, 59, 83)$ where:

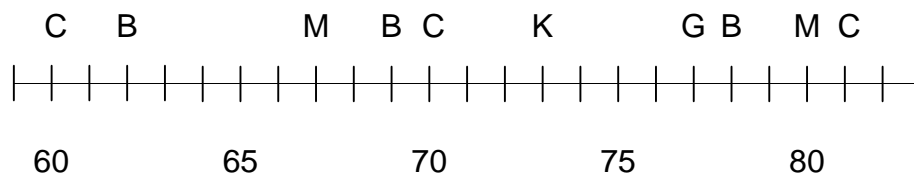


FIGURE 6.1: A Graphical Representation of the Sequence of Events s

An examination of the event sequence in Figure 6.1 shows that it started at time 59 and terminated at 83. The sequence s has 10 events that all occurred in the time interval $[59; 83]$.

Time Window

Since the reason behind using the event sequence abstraction is to identify and calculate episode frequency for each given class of episodes, one needs to establish a time interval over which such frequency is defined. Mannila *et al.* [120] define a time window as a slice of an event sequence as exemplified in Figure 6.2 (time window between 26 and 31 events). As a result, it is clear to see that the event sequence is made up of a number of consecutive time windows. As far as is concerned to the size of such time window, it is up to the user to establish it sufficiently large enough for events to

occur within it. Note that the choice of an appropriate window is important as it impacts directly on the value for such frequencies.

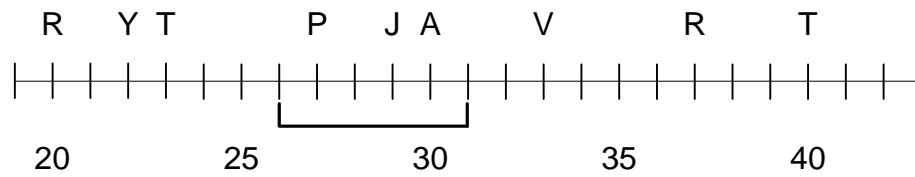


FIGURE 6.2: Time Window within Sequence s

Formally, a time window within an event sequence $s = (s, T_s, T_e)$ is seen as a sequence of events $w = (w, t_s, t_e)$, with $t_s < T_e$ and $t_e > T_s$, and w consist of the pair (A, t) from s where $t_s \leq t < t_e$. The time difference $t_s - t_e$ is defined as the window size of w and is represented by width (w). As a result, given the event sequence s and the integer win , the set of all Windows w of size win in this sequence s are denoted as $W(s, win)$. Further, the first and last windows reach out of the sequence timeline, in that the first one only contains the first time sequence element whereas the last one contains only the last point in time.

Episodes

Episodes are defined as partially ordered events according to their occurrence in time. On a more formal tone, an episode is seen as (V, \leq, m) , where V is the set of all nodes in the episode, the notation \leq symbolizes the events order in time within an episode, and m maps ($m : V \rightarrow Seq$) the nodes V to their respective events within the sequence.

Episodes fall into the following three classes: parallel, serial e and non-parallel and non-serial, as illustrated in Figure 6.3.

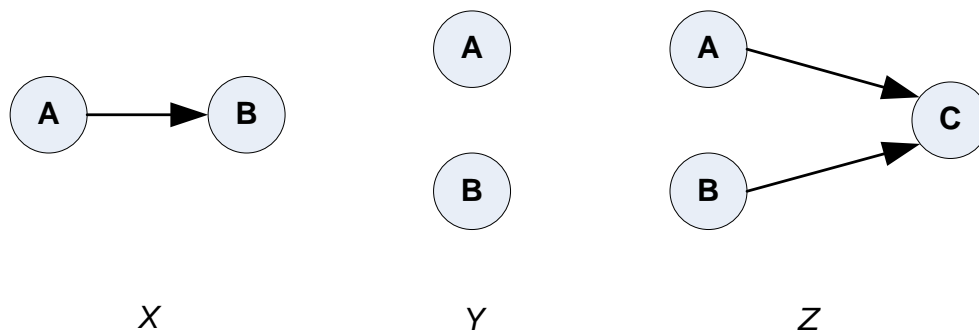


FIGURE 6.3: X, Y and Z represent serial, parallel and non-serial non-parallel episodes

Episode X is said to be serial if subsequently all events of types A and B happen in a given order, in other words, their order is important within a sequence of events. Formally, A and $B \in V$, $A \leq B \neq B \leq A$ if $A \neq B$ for all A and B within the episode X . Episode Y is described as parallel when it does not enforce a strict restriction in the order of its events, turning their temporal order irrelevant. Formally, Y is such that for A and $B \in V$, with $A \neq B$. The third episode Z , from the figure, is called non-parallel and non-serial at the same time. This is the case when events A and B are both part the same episode and have no temporal relationship between them whatsoever.

Sub-episodes

Sub-episodes are based on the idea of that one episode may contain others embedded within it. For example, a look at Figure 6.3 shows that Y is sub-episode of Z , as there is

mapping m that connects nodes A and B with others. To put it more simply, both nodes of Y have corresponding ones in Z .

Formally, $Y = (V', \leq', m')$ is a sub-episode of $Z = (V, \leq, m)$, written as $Y \leq Z$, as there exists a mapping $f: V' \rightarrow V$ such that $m'(v) = m(f(v))$ for all $v \in V'$ and for all $v, w \in V'$ with $v \leq w$ and also $f(v) \leq f(w)$.

Episode Occurrence

An episode is said to occur within a sequence if and only if all its events happen in such sequence while all partial order is maintained.

Formally, an episode $X = (V, \leq, m)$ happens within an event sequence $s = \langle (A_1, T_1), (A_2, T_2), \dots, (A_n, T_n) \rangle$ if there exists a mapping represented by the function $f: V \rightarrow \{1, 2, 3, \dots, n\}$ for all the nodes from X to events in s such that $m(x) = A_{f(x)}$ for any $x \in V$, and $x, y \in V$ with $x \neq y$ and $x \leq y$, hence one must also verify that $t_{f(x)} < t_{f(y)}$.

Episode Frequency Calculation

The frequency of an episode is defined by Mannila *et al.* [120] as the fraction of windows in which a given episode takes place. Hence for an event sequence s and an episode window of size win , the frequency of an episode E within s is given by the formula:

$$fr(E, s, win) = \frac{|\{\mathbf{w} \in W(s, win) | E \text{ occurs in } \mathbf{w}\}|}{|W(s, win)|}$$

To establish whether an episode E is considered frequent or not a threshold (min_fr) is used. As a result, E is said to be frequent if when its frequency $fr(E, s, win) \geq min_fr$. The set of all frequent episodes for a sequence s is given by $\mathcal{F}(s, win, min_fr)$. An important observation to be made here is that whenever an episode is considered as frequent, then also all its sub-episodes are. This is an important result, as seen later, used for the reduction of candidate generation for frequent episodes.

Discovering Episode Rules

Once frequent episodes are determined, these are then used in the study of event correlations. Such relationships are known as episode rules. A rule between two episodes X and Y is formally defined as $X \Rightarrow Y$, called R_{xy} , if X is a sub-episode of Y . For example, the episodes $(A \rightarrow B)$ and $(A \rightarrow B \rightarrow C)$ are frequent, with the frequencies f_1 and f_2 respectively. A resulting rule is $(A \rightarrow B) \Rightarrow (A \rightarrow B \rightarrow C)$ if the expectation for $\left(\frac{f_2}{f_1}\right)$ is higher than an established threshold.

The confidence of a rule is the ratio of its sub-episode expectation by that of the episode. In other words, it represents the conditional probability for Y taking place within a window, given that the episode X did happen in the same window.

Mannila *et al.* [120] suggest two approaches for calculating episode frequencies: the first one is based on the number of windows whereas the other is based on minimal occurrences. In this work, the first one is used for simplicity.

6.1.2. Related Work

Many data mining approaches have been recently applied to design of collaborative IDSs, and frequent episode is one of these. First proposed by Mannila *et al.* [120], there has been several studies devoted to applying frequent episodes in the designs of NIDSs [181][182][183][184][185]. Lee *et al.* [181] developed a data-mining framework as core to an IDS based on the use of the association rules and frequent episodes. They modified Mannila *et al.*'s algorithms to use axis attribute(s) and reference attribute(s) as forms of item constraints, intended to compute only the relevant episodes (thus, ignore non-relevant one). In addition, Lee *et al.* used an iterative level-wise approximate mining procedure to uncover low frequency, but important, episodes. Luo and Bridges [182] imposed fuzzy logic to frequent episodes mining. This way, flexible episodes could be mined, thereby enhancing the detection performance of the proposed NIDSs. Luo *et al.* [183] modified the method of Luo and Bridges [182] in order that the proposed system could be applied in "near" real-time detection. However, like Luo *et al.* [183], the actual response time on an attack was not explicitly reported, and only a single attack, named mscan, was experimented.

Qin and Hwang [184] proposed a new Internet trace technique for generating frequent episode rules to characterize Internet traffic events. These episode rules were used to distinguish anomalous sequences of TCP, UDP, or ICMP connections from normal traffic episodes. In addition, fundamental pruning techniques were introduced to reduce the rule search space by 70% when analyzing the DARPA 1999 traffic Dataset. Hwang *et al.* [185] proposed a hybrid system, which combined the advantages of the low false-positive rate of a signature-based intrusion detection system with the ability of anomaly detection system (ADS) to detect novel unknown attacks. By mining anomalous traffic episodes from Internet connections, Hwang *et al.* [185] built an ADS that detects anomalies beyond the capabilities of the signature-based Snort system. A weighted signature generation scheme was developed to integrate ADS with Snort by extracting signatures from detected anomalies. That is, the hybrid system extracts signatures from the output of ADS and adds them into the Snort signature database for almost immediate and accurate intrusion detection. Experiments on real-world audit data showed that the rate detection of HIDS is equal to 60%, when the results using Snort and BRO system are 30% and 22% respectively.

Soleimani and Ghorbani [186] introduced changes in Mannila *et al.*'s algorithms to manipulate the large quantity of alerts issues by IDSs. The main focus of this work was given to the discovery of all possible alert sequences and their combinations. The most critical among these were then identified. When applying the strategy to the DataSet LLDDoS 1.0 from DARPA collected in 2000, good results were obtained as far as the detection of critical intrusions, memory usage and execution time. The gain shows as much as 90% reductions in alerts, even though critical attacks were still identified and those occurring in more than one place were maintaining.

6.2. FER Analyzer: Design and Implementation

Core to this important module is to correlate alerts and increases the detection accuracy. For this reason, its architectural design is modular to allow ease future modification and the seamless addition of new components. Figure 6.4 depicts a functional diagram of the architectural components in accordance with their roles.

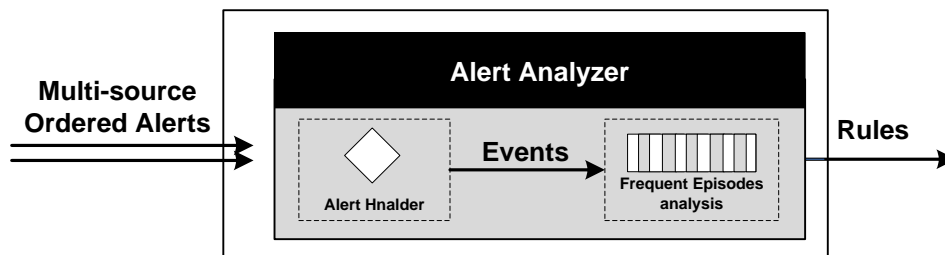


FIGURE 6.4: Functional diagram of FER Analyzer

6.2.1. Alert Handler module

The alert handler module performs alert translation into internal format (values from a finite alphabet starting with A) supported for analysis.

All received alerts are correlated (translated) to an event type in preparation for establishing frequent episode discovery. The selected and extracted attributes of each alert are: class or type of attack, IP address and port number of source system or host, and IP address and port number of target system or host. A correspondence table between events and alerts attributes is built as shown in Table 6.1²⁶.

TABLE 6.1: Example of event types, event names and their attributes

Event Type	Event Name	Source IP	Destination IP
A	Community SIP DNS no such name treshold	172.16.112.100	172.16.115.20
B	NETBIOS NT NULL session	172.16.112.100	172.16.112.20
C	ATTACK-RESPONSE directory listing	172.16.112.194	172.16.112.100
D	NETBIOS NT NULL session	172.16.112.20	172.16.112.100
E	ATTACK-RESPONSE Invalid URL	172.16.113.148	207.200.75.201
F	WEB-MISC RBS ISP /newuser access	134.205.131.13	172.16.117.52
G	WEB-MISC /doc/access	172.16.112.100	135.13.216.191
H	ATTACK-RESPONSE 403 Forbidden	172.16.113.204	137.245.85.134

Other attribute selected and extracted of each alert is Timestamp. It is used in association with Table 6.1 to build a list of formatted events time ordered (in seconds), as shown in Figure 6.5.

²⁶ The events (alerts) presented in Table 6.1 are extracted from DARPA 2000 dataset, LLDOS 1.0 inside scenario.

1	A I
2	C T F C
3	B O
7	A X K M
8	A O
9	B
10	C K P D
11	B O
13	C M O V
14	A Y U
15	A L P J
19	A S H U L R J I
25	B
27	A

FIGURE 6.5: Event sequence representation

Once completed, this list may then be sent to frequent episodes analysis.

6.2.2. Frequent Episode Analysis module

The present Frequent Episode Analysis (FEA) module is used to generate much rules needed to indicate the eminent occurrence of attacks as a result of observing some known sequences of alerts.

Recall that the parameters window size and threshold are important for the precision of the results. The processing overhead is not a concern as this processing may be performed offline and periodically to retrain the FEA module. The analysis made by Frequent Episode Analysis module can be divided in four activities: event collector, candidate generator, generator of frequent episodes and rules generator.

Event Collector

Event collector scans the list with the event types (Figure 6.5) and identifies those that are more frequent. This process is described in Algorithm 6.1 [120]. Basically, it receives a list of episodes of the same size as input and checks which among these are frequent. It verifies if each episode is contained in the global event sequence as shown by the code in line 5. As a result, all frequent episodes are returned.

Algorithm 6.1 Simplified algorithm for checking frequent episodes in the list of events

Input: episodesVector with size **tam**

```

01: frequents = []
02: for all episode in episodesVector do
03:     eventList = getEventSequence(E)
04:     if episodio.hasIn(eventList) then
05:         frequents += episode
06:     end if
07: end for
08: return frequents

```

In the algorithm, an episode $\alpha = (V, \leq, m)$ is represented as a lexicographically sorted array of event types.

Candidate Generator

Candidate generator receives a list of frequent episodes of size X and generates a new list with possible frequent episodes of size $X+1$. The Algorithm 6.2 [120] describes the candidates generation for serial episodes. This calculation demands a careful design since is crucial to turn efficient the search for frequent episodes considering that the number of possible frequent episodes grows exponentially with the increase of the window size.

Algorithm 6.2 Simplified algorithm for candidate generation

Input: frequent episodes **FrequentEpisodes**

```

01: candidatesWithSizePlus1 = []
02: for all episode in FrequentEpisodes do
03:   possibleCandidates = episodio.getChilds( )
04:   for all candidate in possibleCandidates do
05:     subsets = candidate.getAllSubSets()
06:     if (FrequentEpisodes.isSubSet(subsets) == TRUE) then
07:       candidatesWithSizePlus1 += candidate
08:     end if
09:   end for
10: end for
11: return candidatesWithSizePlus1

```

The algorithm receives as input a ordered list of frequent episodes of size X . For any given input frequent episode, all possible frequent episode candidates are calculated. However, it is important to emphasize that if an episode is frequent in an event sequence, then all subepisodes are also frequent (FER lemma 1 [120]). In order to attend this rule, each episode of frequent episodes list (line 2) must have its possible childs with size plus 1 verified (line 3). Consequently, each one of these possible candidates (line 4) is used to generate all possible subsets with size equal to the episode (line 5). To qualify as a new frequency episode of size $X+1$, it must frequent, i.e., all its sub-episodes of size X must be present in the input frequency list as enforced in line 6. If the result is true, then the candidate is add to the list of candidates with size plus 1 (line 7). The algorithm finishes by returning an ordered list of candidate serial episodes of size $+1$.

In order to provide a better understanding of this algorithm, a simple example is explained. Given a frequent episodes list composed by four elements $\{AA, AB, AC, AD\}$, the possible candidates of the episode AB will be $(ABA, ABB, ABC, ABD, \dots, ABZ)$ and each one of them will be tested to generate possible candidates. Evaluating the first candidate, ABA , the subsets originated from it are AB, BA and AA . So, to prove that the candidate ABA is frequent, its subsets $(AB, BA$ and $AA)$ are matched with the frequent episodes list. As result, ABA is a possible candidate since AB, BA and AA are represented in frequent episodes list by the episodes AA and AB . On the other hand, the candidate ABZ is not since the subsets AZ and BZ not have representation of the frequent episodes list.

Generator of Frequent Episodes

A generator of frequent episodes can be seen as the driver for the previous two modules. It calculates all frequent episodes of all sizes using the output from the candidate generator and event collector. Algorithm 6.3 [120] describes how to calculate a collection of frequent episodes from an event sequence E of episodes.

Algorithm 6.3 Simplified algorithm for calculating frequent episodes

Input: event sequence **E**, window size **win** and frequency **fr**

```

01:  $C_1 = \{ \text{all elements in } E \text{ with size equal to } 1 \}$ 
02:  $candidateSize = 1$ 
03:  $candidateVector = \{ \text{all element } \in C_1 \text{ and } |element| == 1 \}$ 
05: for  $candidateSize$  to  $win$  do
06:   /* Checking for frequent episodes (Algorithm 6.1) */
06:    $FrequentEpisodes = checkFrequentEpisodesInEventList(candidateVector, win, fr)$ 
07:    $AllFrequentEpisodes += FrequentEpisodes$ 
08:    $candidateSize++$ 
09:   /* Candidate generation (Algorithm 6.2) */
10:    $candidateVector = generateCandidates(FrequentEpisodes)$ 
11: end for
12: return  $AllFrequentEpisodes$ 

```

It starts with the definition of a set (C_1) containing all elements from event sequence E with size equal to 1 (line 1), a control variable $candidateSize$ to permit computing frequent episodes according to the window size win (line 2) and a structure $candidateVector$ to receive all generated candidates to be frequent episodes (line 3). To calculate the frequent episodes, the algorithm 6.3 keeps running until it achieves the window size limit (line 5). On each iteration, the algorithm first verifies the frequency of the candidate episodes from the event sequence (line 6) calling the algorithm 6.1. As result, the returned frequent episodes are store in a general list $AllFrequentEpisodes$ (line 7) and the $candidateSize$ variable is incremented by one. Secondly, it calls the algorithm 6.2 to generate the candidates for frequent episodes, returning all possible candidates episodes with size incremented by one (line 10). The algorithm finishes when returning all frequent episodes.

Rule Generator

A rule generator extracts the rules one is seeking. It notifies an application such as an IDS, how likely an attack or an anomaly is underway. Three main advantages are important to mention here: considerable reduction of alert messages; higher precision and confidence in alerts; and the prediction of attacks and anomalies. The pseudo-code presented in Algorithm 6.4 [120] is responsible for rule calculation.

Algorithm 6.4 Simplified algorithm for rule calculation

Input: event sequence **E**, window size **win**, frequency **fr** and confidence **conf**

```

01:  $rules = []$ 
02: /* Find frequent episodes (Algorithm 6.3) */
03:  $FrequentEpisodes = calculateFrequentEpisodes(E, win, fr)$ 
04: /* Generate rules */
05: for all  $episode$  in  $FrequentEpisodes$  do
06:   for all  $subepisode$  in  $episode$  do
07:     if  $frequency(episode)/frequency(subepisode) \geq conf$  then
08:        $rules += [episode, subepisode, confidence(episode/subepisode)]$ 
09:     end if
10:   end for
11: end for
12: return  $rules$ 

```

This simple algorithm first starts calling the algorithm 6.3 (calculateFrequentEpisodes) to calculate frequent episodes (line 3) for a given event sequence E , when using a window of size win and a frequency fr . As result, a list of all frequent episodes is returned (*FrequentEpisodes*). Secondly, the algorithm performs the rule calculation process through a series of iterations. The first one extracts episodes that composes a list of frequent episodes (line 5). The second one extracts subepisodes relatively to the parent episode (line 6). The extraction of subepisodes is based on the FER lemma 1 [120] that says that if an episode is frequent in an event sequence, then all subepisodes are also frequent. Next, with the episode and its *subepisodes* at hand, the algorithm 6.4 tests if the relation (proportion) between an *episode* and its *subepisode* is greater than or equal to a defined confidence threshold (line 7). If the result is true, then a new *rule* is generated as shown at line 8. The algorithm finishes returning all generated rules.

Rule Reduction

Although functional and essential to frequent episodes analysis, the calculation and generation of rules typically results in a huge number of FER and consequently a high number of redundant or repeated rules. In order to solve this inefficiency, algorithm 6.5 employs two pruning techniques, proposed by Qin and Hwang [184], to reduce the rule space and to provide a simplified view of data patterns. The idea is to establish if an FER is effective (more frequently used) or ineffective (rarely used).

Algorithm 6.5 Simplified algorithm for rule calculation

Input: rules r

```

01: reducedRules = []; newRules = [];
02: for all rule in r do
03: /* Application of Transposition Law */
04:   newRules += TranspositionReduction(rule);
05: end for
06: for all rule in newRules do
07: /* Application of Elimination of Redundant Law */
08:   reducedRules += EliminationRedundant(rule);
09: end for
10: return reducedRules

```

The first law, transposition, asserts that given these two FERs ($A \rightarrow AAAA$ and $A \rightarrow AAAA$), which describes behaviors for event A , the first one is seen as being more effective than the second one. This is because of its satisfaction of the transposition law, that is, the second rule can be induced by the first one. Therefore, the first rule is kept (effective) and the second one is removed. In general lines, the goal is to make the left hand side (LHS) as short as possible due to the fact that shorter rules are often easier to apply or to compare.

The elimination of redundant law also assumes that rules with shorter LHSs are more effective than rules with longer LHSs. This way, if there are two FERs ($A \rightarrow B$ and $B \rightarrow C$) in the rule set and there is a very frequent rule ($A \rightarrow BC$), it corrects to assume that the rule ($A \rightarrow BC$) is redundant, since it can be reconstructed from the two previous rules. Therefore, the two rules are kept (effective) and the last one is removed.

The result of algorithm 6.5 are rules without redundant elements. Figure 6.6 illustrates an example of rules reduction process.

Rule 136:	M1 -----> M1AAAAAAAAAAAAA with confidence 0,81
Rule 137:	M1A -----> M1AAAAAAAAAAAAA with confidence 0,91
Rule 138:	M1AA -----> M1AAAAAAAAAAAAA with confidence 0,91
Rule 139:	M1AAA -----> M1AAAAAAAAAAAAA with confidence 0,91
Rule 140:	M1AAAA -----> M1AAAAAAAAAAAAA with confidence 0,91
Rule 141:	M1AAAAA -----> M1AAAAAAAAAAAAA with confidence 0,91
Rule 142:	M1AAAAAA -----> M1AAAAAAAAAAAAA with confidence 0,91
Rule 143:	M1AAAAAAA -----> M1AAAAAAAAAAAAA with confidence 0,91
Rule 144:	M1AAAAAAAA -----> M1AAAAAAAAAAAAA with confidence 0,91
Rule 145:	M1AAAAAAAAA -----> M1AAAAAAAAAAAAA with confidence 0,91
Rule 146:	M1AAAAAAAAAA -----> M1AAAAAAAAAAAAA with confidence 0,91
Rule 147:	M1AAAAAAAAAAA -----> M1AAAAAAAAAAAAA with confidence 0,91

FIGURE 6.6: Normal rules and reduced rule for a *sadmind* request in LLDOS 1.0 Outside

In Figure 6.6, the event A indicates a port scan over the network and M1 an attempt to obtain root privileges through *sadmind*. Since the probability of the occurrence of *sadmind* attacks be followed by port scans is constant (same confidence threshold), applying the transposition law, one generate a single rule to express these attack situation (Rule 39), containing the lower LHS possible and the higher confidence. In this example, 12 rules are reduced in one.

6.2.3. Implementation

The FER Analyzer module was entirely developed in Java (Version 1.6.0), using Eclipse (version 3.3.2) as IDE.

6.3. Evaluation

This section describes a series of experiments in order to evaluate the performance and results of FER Analyzer.

All experiments were conducted using a computer AMD Athlon 64 3000+ processor, with 4 GB main memory and 500 GB of hard disk, under the Ubuntu 10.4 Linux operating system.

6.3.1. Performance overview

In order to test the performance of the present implementation, the DARPA 2000 dataset [179] (LLDOS 1.0 and LLDOS 2.0.2 scenarios) was first used to measure the influence of window size and frequency threshold in the frequent episodes generation.

DARPA 2000

The LLDOS 1.0 scenario consists of a sequence of 1109 alerts (inside) and 2465 alerts (outside) covering a time period of almost 3 hours. Considering that each alert is composed by the following five-tuple (source IP address, source port, destination IP address, destination port and class of attack), there are 602 and 90 different types of events, respectively, with very diverse frequencies and distributions. On average, there is an alert every minute. However, since the scenarios illustrate attacks, the alerts tend to occur in bursts. For example, 169 alerts occurred in a period of one second.

Tables 6.2 and 6.3 represent performance statistic for finding frequent episodes in LLDOS 1.0 *inside* and *outside* scenarios with different window sizes and frequency threshold of 0.005. The time required, the number of episodes found, the number of possible candidates and the level of participation (frequent episodes/candidates rate) are also presented.

TABLE 6.2: Performance for LLDOS 1.0 inside scenario

Window Size (s)	Candidates	Frequent Episodes	Level of Participation (%)	Time (s)
2	0	0	---	4.8
3	1	1	100.00%	5.3
4	9	3	33.33%	5.9
5	100	10	10.00%	6.5
6	100	10	10.00%	7.3
7	100	11	11.00%	7.6
8	100	11	11.00%	8.0
9	100	11	11.00%	8.4
10	144	13	9.03%	8.8
11	144	13	9.03%	9.2
12	146	15	10.27%	9.5
13	152	21	13.82%	9.9
14	152	21	13.82%	9.9
15	152	21	13.82%	10.5
16	152	21	13.82%	11.0
17	152	21	13.82%	11.3
18	152	21	13.82%	11.7
19	152	21	13.82%	12.0
20	181	24	13.26%	12.5

TABLE 6.3: Performance for LLDOS 1.0 outside scenario

Window Size (s)	Candidates	Frequent Episodes	Level of Participation (%)	Time (s)
2	2	2	100.00%	1.0
3	9	5	55.56%	1.3
4	18	7	38.89%	1.7
5	199	18	9.05%	4.5
6	200	19	9.50%	5.0
7	201	20	9.95%	5.2
8	202	21	10.40%	5.7
9	203	22	10.84%	6.0
10	204	23	11.27%	6.2
11	199	24	12.06%	6.7

12	205	25	12.20%	7.0
13	206	26	12.62%	7.2
14	218	37	16.97%	7.8
15	248	39	15.73%	8.8
16	280	57	20.36%	9.7
17	286	59	20.63%	10.4
18	288	61	21.18%	10.7
19	291	64	21.99%	11.5
20	394	69	17.51%	14.5

In Tables 6.2 and 6.3, it is possible to see clearly the window size influence over the number of frequent episodes obtained. Since the discovery of frequent episodes is related with the existence of possible candidates, it is correct to assert that as more the window size increases, the greater will be the number of candidates and consequently the number of frequent episodes discovered. The processing time and the level of participation also increases the same way. Regarding the stagnation on the number of candidates and the frequent episodes (Table 6.2, window sizes 13 to 19), such fact happens because it is not possible to generate new candidates for these window sizes. In this specific case, all possible candidates are totally composed by episode candidates with size 2 (145 candidates) and 3 (8 candidates).

Figure 6.7 represents the effect of the window size on the number of frequent episodes for both scenarios.

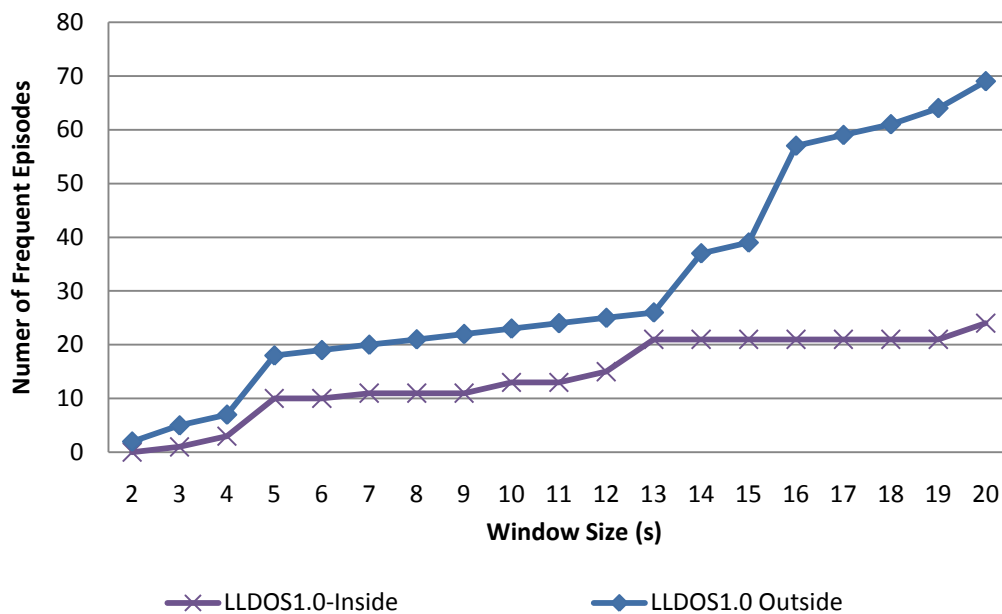


FIGURE 6.7: Number of frequent episodes as a function of window size, with frequency threshold of 0.002, for LLDOS 1.0 scenarios

The difference among the number of frequent episodes into *inside* and *outside* scenarios is explained by the fact that the DDoS attack generated in these scenarios originated a large amount of packets with random originating addresses. Hence, the thousands of triggered alerts, with different source addresses, did not correlate with each other's. Consequently, they were not potentially repeated which limited frequent episodes. An examination of the correspondence table (Alert Handler module) revealed

that more than 98% of the alerts taking place with the *outside* case were not repeated within the internal scenario.

Within the LLDOS 2.0.2 scenario, there are 935 alerts (*inside*) and 1108 alerts (*outside*) covering a time period of almost 1 hour and 30 minutes, where there are 867 and 41 different types of events, respectively, with very diverse frequencies and distributions. Performance statistics of LLDOS 2.0.2 for the *inside* and *outside* scenarios are available in Tables 6.4 and 6.5.

TABLE 6.4: Performance for LLDOS 2.0.2 inside scenario

Window Size (s)	Candidates	Frequent Episodes	Level of Participation (%)	Time (s)
2	2	2	100.00%	4.3
3	2	2	100.00%	4.8
4	15	7	46.67%	5.7
5	40	14	35.00%	6.5
6	40	14	35.00%	7.0
7	41	15	36.59%	7.5
8	43	17	39.53%	7.8
9	43	17	39.53%	8.3
10	56	20	35.71%	8.8
11	56	20	35.71%	9.2
12	56	20	35.71%	9.7
13	56	20	35.71%	10.1
14	56	20	35.71%	10.5
15	128	30	23.44%	11.0
16	128	30	23.44%	11.0
17	128	30	23.44%	11.3
18	128	30	23.44%	11.7
19	128	30	23.44%	12.8
20	130	32	24.62%	13.3

TABLE 6.5: Performance for LLDOS 2.0.2 outside scenario

Window Size (s)	Candidates	Frequent Episodes	Level of Participation (%)	Time (s)
2	3	3	100.00%	0.4
3	4	4	100.00%	0.5
4	6	5	83.33%	0.5
5	12	7	58.33%	0.7
6	20	9	45.00%	0.7
7	21	10	47.62%	0.8
8	22	11	50.00%	1.1
9	23	12	52.17%	1.2

10	24	13	54.17%	1.3
11	34	15	44.12%	1.5
12	35	16	45.71%	1.6
13	36	17	47.22%	1.6
14	37	18	48.65%	1.8
15	39	20	51.28%	2.0
16	116	27	23.28%	3.6
17	118	29	23.28%	3.6
18	121	32	24.58%	4.1
19	128	39	30.47%	4.3
20	157	52	33.12%	4.6

Figure 6.8 illustrates the effect of the window size on the number of frequent episodes for both LLDOS 2.0.2 scenarios.

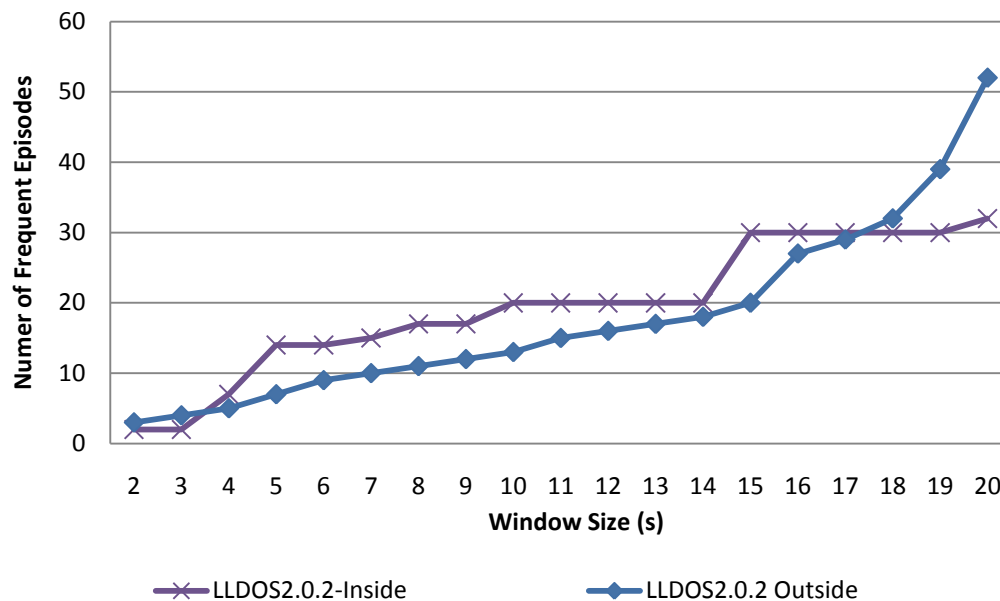


FIGURE 6.8: Number of frequent episodes as a function of window size, with frequency threshold 0.002, for LLDOS 2.0.2 scenarios

Before showing the results of episodes rules, it is important and necessary to clarify the importance and use of the frequency threshold parameter in any FER analysis. The concept of frequency, represented by the picture of frequency threshold, is essential to discover all frequent episodes of a sequence and consequently to obtain rules that describes connections between events. This way, typical values of frequency threshold for analysis are: 0.001, 0.002, 0.005, 0.01, 0.02, 0.05 and 0.1 [120].

In fact, most works using FER employ frequency threshold values equal or superior to 0.01. The explanation is simple. Small values of frequency threshold such as 0.001 and 0.002 allow the generation of a huge amount of candidates and frequent episodes, with relevant impacts on processing time. For example, in LLDOS2.0.2 *inside*, the use of frequency threshold value of 0.001, with window size equal to 12, generates 753.424 possible candidates of size 2, during a period of 25 minutes of processing. In LLDOS 1.0 *outside*, window size superior to 8 generates 857.000 possible candidates of size 2.

Nonetheless, values above 0.02 practically inhibit the generation of candidates since only massive events (like massive DDoS attacks) can be recognized as frequent. In the present four scenarios, no frequent episode was detected with this value.

6.3.2. Episode Rules

In order to test episode rules generation (the main goal of FER analysis), the current implementation establishes two FER parameters: window size and confidence level. The first one has influence over the number of frequent episodes generated and consequently in the number of rules. The second one impacts the quality of such rules. The higher the confidence level, the better is the quality and reliability of the obtained rules.

Figures 6.9, 6.10, 6.11 and 6.12 plot the rule set growth against different window size parameters four DARPA 2000 dataset scenarios. For clarity, the confidence level of 0.6 and the frequency threshold of 0.005 are assumed in all analysis.

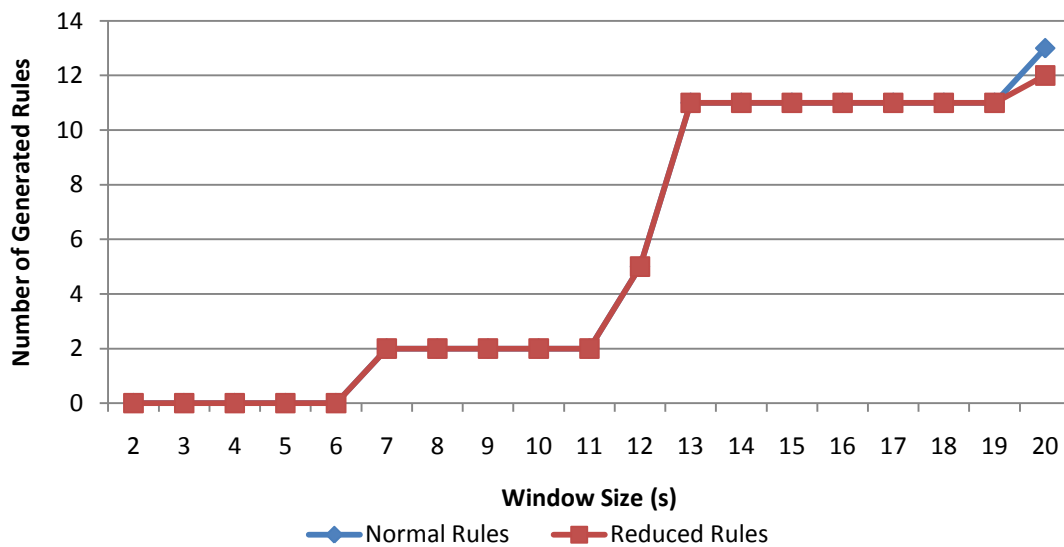


FIGURE 6.9: Rule space generated from LLDOS 1.0 Inside

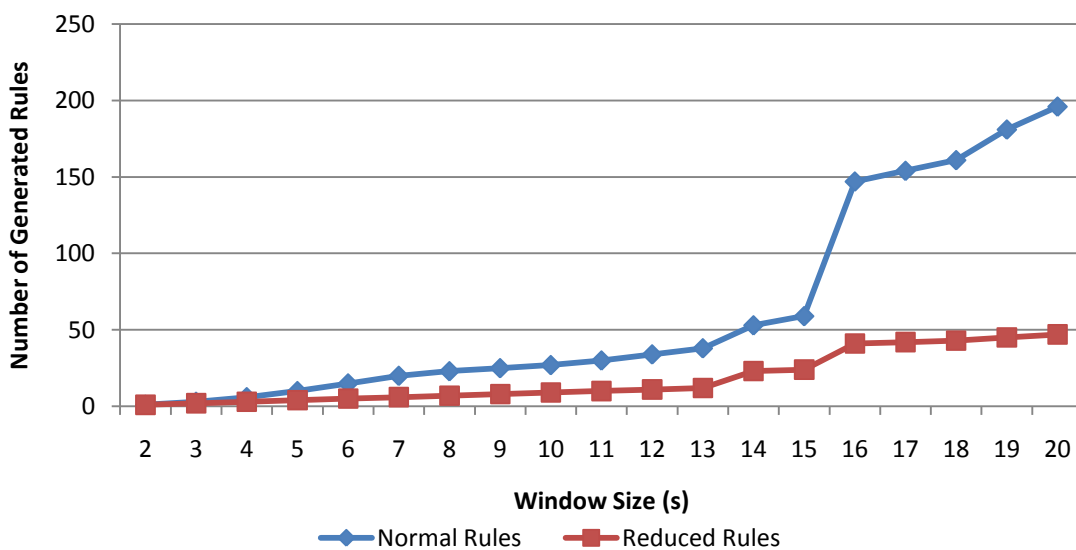


FIGURE 6.10: Rule space generated from LLDOS 1.0 Outside

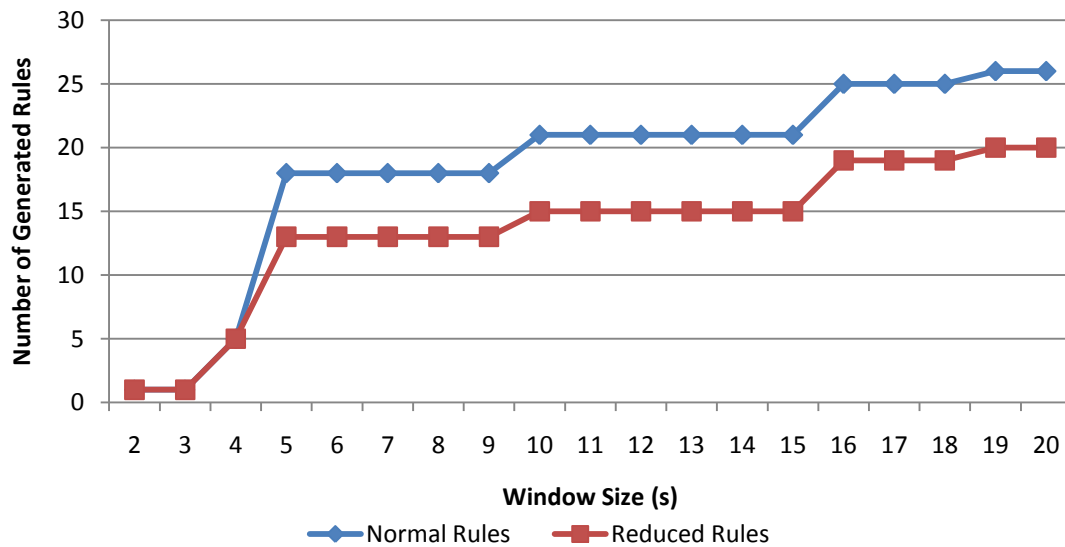


FIGURE 6.11: Rule space generated from LLDOS 2.0.2 Inside

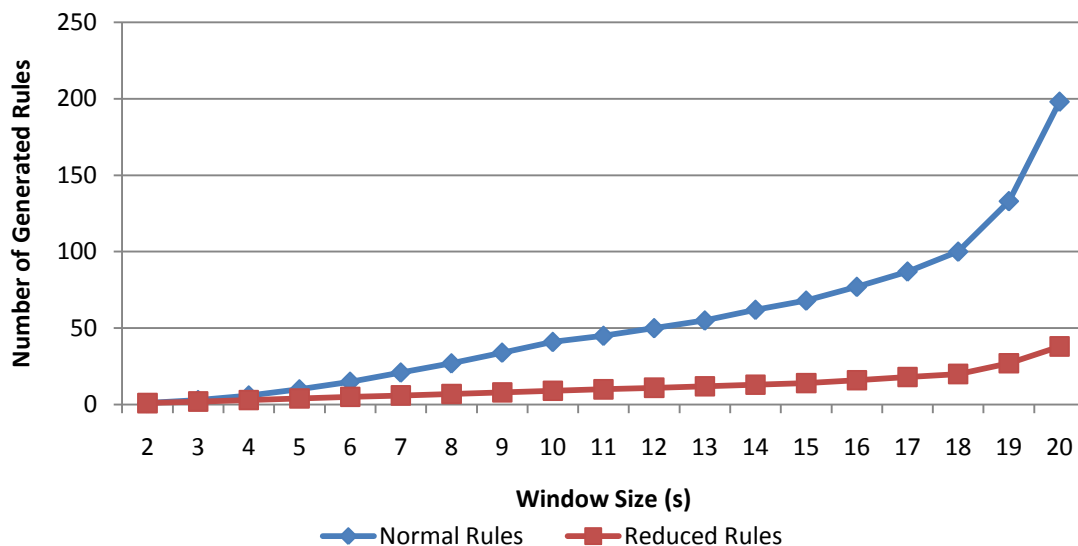


FIGURE 6.12: Rule space generated from LLDOS 2.0.2 Outside

A relationship between rules and window size is shown where the difference between normal and reduced rules generated increases as the window size also increases, although this reduction had been minimal in both inside scenarios. LLDOS 1.0 *inside* had a reduction of 8% (Figure 6.9) whereas LLDOS 2.0.2 *inside* had a reduction varying between 24 and 29% (Figure 6.11). Such fact is explained by the features of the traces, which generate many events. Consequently, this huge diversity (602 events of 1109 alerts in LLDOS 1.0 and 867 events of 935 alerts in LLDOS 2.0.2) is reflected in a low discovery of frequent episodes and the consequent generation of rules. On the other hand, *outside* scenarios present consistent and relevant reductions. In LLDOS 1.0 *outside*, the variation among rules achieved a range between 33 to 73% (Figure 6.10) when in LLDOS 2.0.2 this variation stayed between 33 and 81% of rules (Figure 6.12).

Figures 6.13, 6.14, 6.15 and 6.16 plot the rule set reduction against different confidence level parameter under four DARPA 2000 dataset scenarios. For clarity of the results, the maximum window size of 20 and frequency threshold of 0.005 were assumed in all analysis.

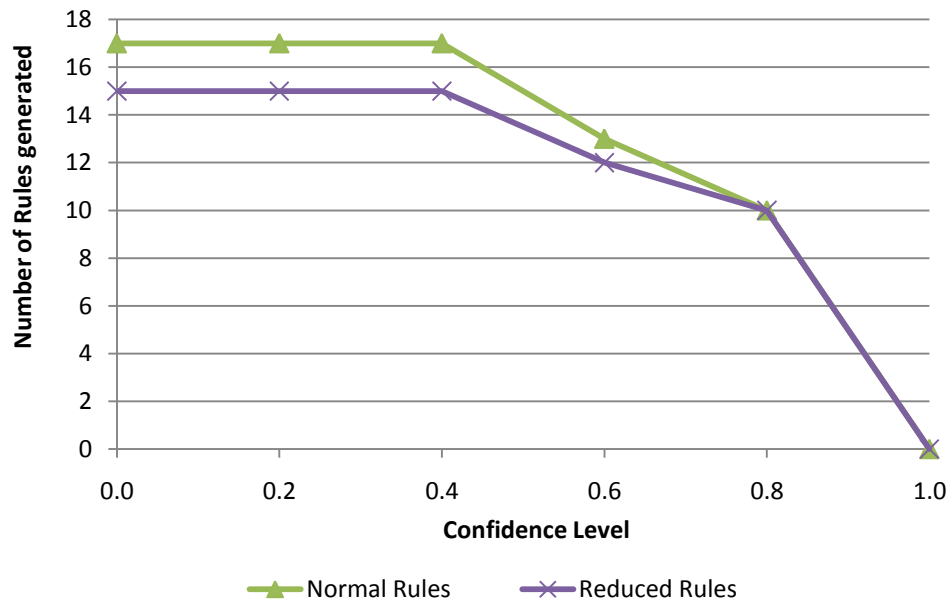


FIGURE 6.13: The effects of pruning for LLDOS 1.0 Inside, with various confidence thresholds, with frequency threshold 0.005 and window size 20

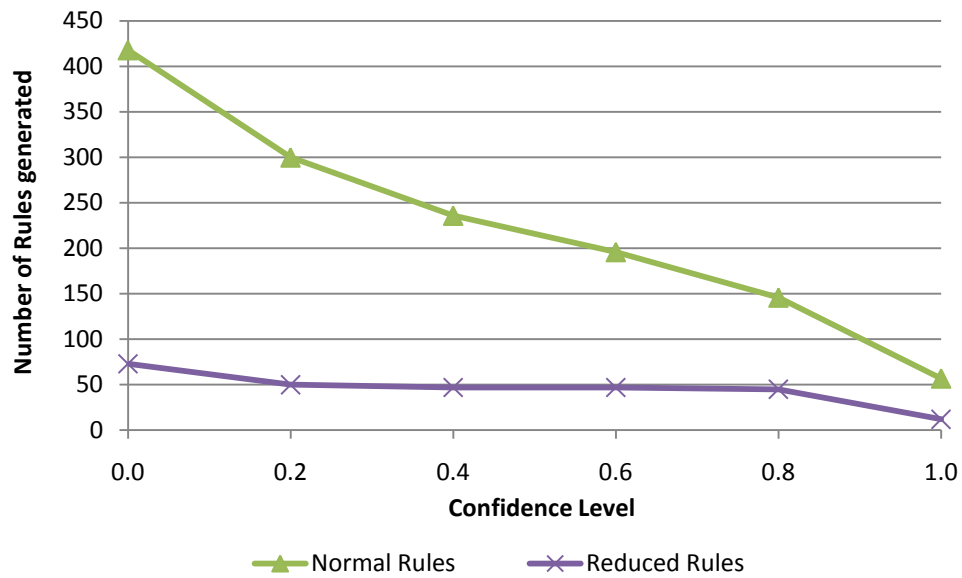


FIGURE 6.14: The effects of pruning for LLDOS 1.0 Outside, with various confidence thresholds, with frequency threshold 0.005 and window size 20

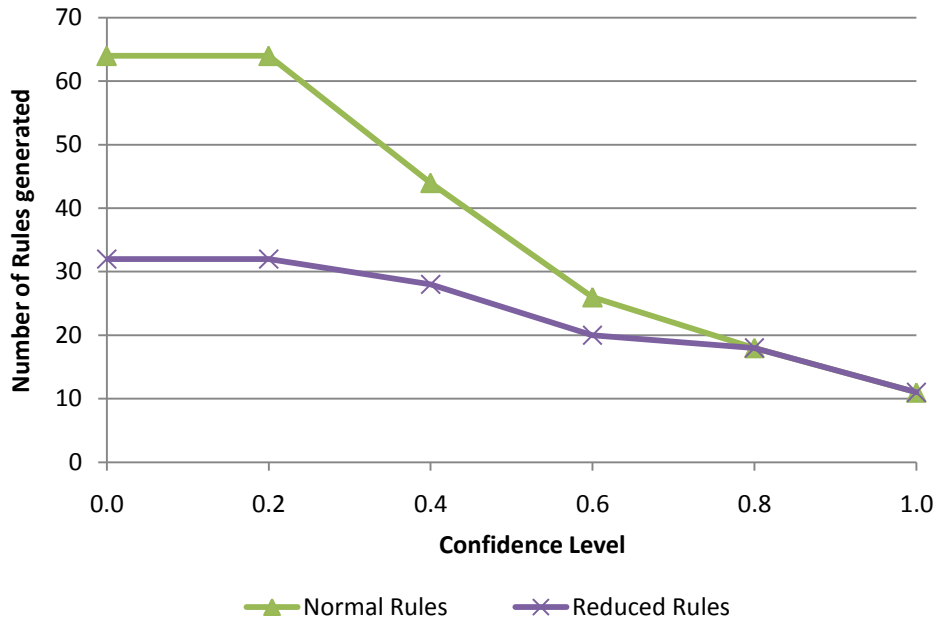


FIGURE 6.15: The effects of pruning for LLDOS 2.0.2 Inside, with various confidence thresholds, with frequency threshold 0.005 and window size 20

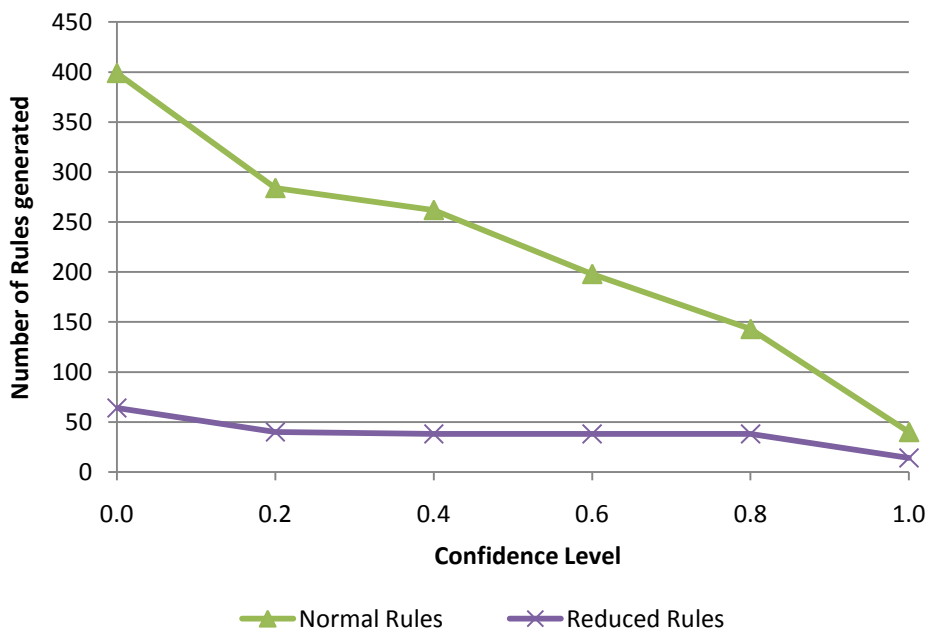


FIGURE 6.16: The effects of pruning for LLDOS 2.0.2 Outside, with various confidence thresholds, with frequency threshold 0.005 and window size 20

Since the rules with a high confidence are often the most interesting and useful ones, especially if they are used for prediction, all Figures show how the number of distinct rules varies as a function of the confidence threshold. From almost 900 rules generated by the four scenarios, 108 have a confidence of exactly 1, for normal rules. For reduced rules, this number is of 37. For many applications it is reasonable to use a fairly low confidence threshold in order to point out the interesting connections, as is discussed in the following subsection.

Applicability of Episode Rules

In order to put in practice the applicability of episode rules, an enforcement agent is built. It translates the episode rules generated by FER Analyzer to firewall rules, specifically for IPTables.

Taking some episodes rules generated for LLDOS1.0 *inside* scenario as example (Figure 6.17), the idea is to translate right hand side of each one in IPTables rules, according to some predefined actions.

...
Rule 6: G -----> GH with confidence 1,00
...
Rule 10: QR -----> QRTU with confidence 1,00
Rule 11: TU -----> QRTU with confidence 1,00
...

FIGURE 6.17: Sample of the episode rules generated for LLDOS 1.0 Inside (confidence thresholds 0.8, frequency threshold 0.002 and window size 20)

The rule 6 represents the relationship between the events G and H. The first is an WEB-MISC /doc/ access alert, with source address 172.16.112.50, source port 44482, destination IP address 172.16.113.204, destination port 80, and severity 2 (medium risk) and the second one is an WEB-MISC finger access alert, with source address 172.16.112.50, source port 33378, destination IP address 172.16.114.50, destination port 80, and severity 2 (medium risk). According to the generated rule, the probability of let event G to be followed by an event H has confidence equal to 1 (100%). This way, the IPTable rules compatible with the episode rule is shown in Figure 6.18.

Rule 6: G -----> GH with confidence 1,00
IPTABLES -A INPUT -i \$INTERFACE -s 172.16.112.50 --dport 80 -p tcp -m limit --limit 300/second -j ACCEPT
IPTABLES -A INPUT -i \$INTERFACE -s 172.16.112.50 --dport 80 -p tcp -m limit --limit 300/second -j ACCEPT

FIGURE 6.18: Translation of an episode rule to IPTable rules

Note that in this example, the action was to limit the malicious source IP address. Such decision can be based on the value of severity parameter. Possible values to the action are: Limit, during 60 seconds for low risk alerts; Extreme-Limit, during 300 seconds for medium risk alerts; and Block (DROP), for high risk alerts.

The rules 10 and 11 represent the relationship between the events Q, R, T and U. All four events have the same class of attack, RPC portmap sadmind request UDP alert, present the same source address 207.77.162.213, same destination port 111 and same severity 2 (medium risk). Q event has source port 54790 and destination IP address 172.16.115.87. R event has source port 54793 and destination IP address 172.16.115.20. T event has source port 55484 and destination IP address 172.16.115.50. U event has source port 55485 and destination IP address 172.16.115.105.

According to the generated rules, the probability of the events Q and R to be followed by the events T and U has confidence equal to 1 (100%). Moreover, the probability of the events T and U to be preceded by the events Q and R has confidence

equal to 1 (100%). This way, the IPTable rules compatible with the episode rules is as follow:

- IPTABLES -A INPUT -i \$INTERFACE -s 207.77.162.213--dport 111 -p tcp -m limit --limit 30/second -j ACCEPT
- IPTABLES -A INPUT -i \$INTERFACE -s 207.77.162.213--dport 111 -p tcp -m limit --limit 30/second -j ACCEPT
- IPTABLES -A INPUT -i \$INTERFACE -s 207.77.162.213--dport 111 -p tcp -m limit --limit 30/second -j ACCEPT
- IPTABLES -A INPUT -i \$INTERFACE -s 207.77.162.213--dport 111 -p tcp -m limit --limit 30/second -j ACCEPT

From LLDOS 2.0.2 *inside* scenario, also using confidence thresholds 0.8, frequency threshold 0.002 and window size 20, the episodes rules generated are showed in Figure 6.19, where H event is an Bad Traffic Loopback IP alert, with source address 127.201.162.238, source port 39965, destination IP address 131.84.1.31, destination port 30906, and severity 3 (high risk) and I event is a Bad Traffic Loopback IP alert, with source address 127.201.162.238, source port 39965, destination IP address 131.84.1.32, destination port 30906, and severity 3 (high risk):

Rule 1:	H -----> HI with confidence 1,00
Rule 2:	I -----> HI with confidence 1,00

FIGURE 6.19: Episode rules generated for LLDOS 2.0.2 Inside (confidence thresholds 0.8, frequency threshold 0.002 and window size 20)

According to the generated rules, the IPTable rules compatible with the episode rules is as follow:

- IPTABLES -A INPUT -i \$INTERFACE -s 207.77.162.213--dport 111 -p tcp -m limit --limit 30/second -j DROP
- IPTABLES -A INPUT -i \$INTERFACE -s 207.77.162.213--dport 111 -p tcp -m limit --limit 30/second -j DROP

In LLDOS 1.0 *outside* and LLDOS 2.0.2 *outside* scenarios, none episode rules were generated.

6.3.3. Real Traffic Analysis

This section presents an evaluation of the present module when submitted to real traffic within a real environment. A LAN segment of the Laboratory GPRT, composed by around 60 computers with two egress links: the first connects the GPRT to the local Internet PoP-PE and the other one to the University Campus network via the IT Nucleus (NTI).

To collect network traffic, Snort version 2.8.3.2 was setup at two gateways, one per external link, and configured with the same IDS rules. Traffic was observed during the days 17 and 19 of November 2009 during 14 hours on the 17th and 18 hours on the 19th.

Frequent Episodes

Unlike the previous scenarios, there is no a priori knowledge of traffic at the laboratory network. Nonetheless, during these two days, around 40.185 alerts were generated. Clearly, a network administrator stands no chance in analyzing these notifications

without the auxiliary use of new tools. Using FER analysis, 348 event types were registered. Table 6.6 contains more data on the GPRT evaluation scenario for frequency threshold equal to 0.001.

TABLE 6.6: GPRT Laboratory Frequency Episode Results.

Window Size (s)	Possible Episodes	Candidates	Frequent Episodes	Level of Participation (%)
1	348	348	11	3.1%
2	82369	121	4	3.3%
3	2.10^7	8	5	62.5%
4	7.10^9	7	6	85.7%
5	2.10^{12}	8	6	75%
6	6.10^{14}	8	7	85.7%
7	2.10^{17}	9	6	66.6%
8	5.10^{19}	8	4	50%
9	1.10^{22}	4	2	50%
10	4.10^{24}	2	1	50%

Overall, the results reinforce the fact that the candidate generation technique lowers substantially the cost of calculating frequent episodes.

Episode Rules

Regarding episode rules, none effective rule was generated. The reason is simple. Although GPRT alerts have produced different frequent episodes, as presented in Table 6.6, the relevance of these frequent episodes is not sufficient to show the connections between events.

For example, employing window size equal to 1 (the greatest number of episodes discovered), only three rules was generated, but all of them with confidence threshold inferior to 0.6. Moreover, as smaller the number of frequent episodes, lower is the number of generated rules.

6.4. Chapter Summary

This chapter presented the design and development of a new module responsible for the increase network security. Based on frequent episodes discovery technique, an Alert Analyzer evaluates multi-source alerts and generates rules, helping on the identification of network attacks and anomalies, increasing the accuracy and decreasing the uncertainty.

In order to generate the frequent episodes rules effectively, the original algorithms proposed by Mannila *et al.* [120] is combined with two episode rules pruning techniques proposed by Qin and Hwang [184]. To evaluate this module, two intrusion data sets were used: DARPA 2000 and one collected from the GPRT research laboratory. As demonstrated, this solution is capable of handling multi-source alert messages, generates effective frequent episodes rules and consequently produce applicable firewall rules.

To sum up, the application of FED analysis to anomaly and attack detections has shown to be useful. The role of the different parameters of such algorithm such as

window size and frequency threshold has been shown through experimentation. The automatic interaction between this module and other security solutions like Firewall to mitigate the identified will certainly increase the effectiveness of this solution and provide the base for an autonomic security management solution.

Chapter 7

OADS Miner: Codename ARAPONGA

The last decade has seen a frightful increase of undesired, unsolicited, often illegitimate Internet traffic, or simply unwanted. Despite being related to spam and denial of service attacks, a great part of this problem directly involves security violations caused by vulnerabilities exploits in software systems and some user services.

To security specialists and many businesses, this has been a welcomed opportunity for the development of tools to combat these threats. To software and hardware manufacturers, this phenomenon requires their constant monitoring of evolving security problems targeting their products. Most of them alert their customers as soon as they discover possible security problems and ask these to apply new patches or undertake complete upgrades. There is nonetheless no way for a software manufacturer to guarantee that its customers are acting comply with its advice. This is often due to many reasons. IT administrators may find themselves overwhelmed with the high number of requests to update their applications, suffering from a lack of human resources, or may simply fail to understand the gravity of a given received alert. The relevance of this kind of solution is witnessed by the existence of dozens of highly visited private and public databases as a well as Web sites about vulnerabilities, anomalies and attacks.

Despite having innumerable Web sites with security information, these often do not coordinate their work, tend to use different report formats with emphasis on varying information, specialize in different types of vulnerabilities, use different severity classifications, lack statistical and event correlation data and may be difficult to search through and visualize. This way, network administrators and IT managers may need to spend precious resources and time browsing, filtering and locating reports on relevant vulnerabilities and security information of interest to them instead of performing other tasks.

In order to address this problem, a Web search system capable to provide integration, into a single place, of security information and automated and advanced search functionality over such content is proposed. This solution is called ARAPONGA.

The rest of this chapter explains how ARAPONGA can be useful in helping users and other systems. Firstly, some background information and related works are presented, aiming to differentiate out this solution from other proposals. Next, an overview design of ARAPONGA is presented, including a detailed description of architectural components and their implementation. An initial evaluation is then presented to validate the developed tool. Lastly, some conclusions are discussed.

7.1. Background and Related Work

The constant human quest for innovation and technological advances resulted in a wealth of data and information scattered, over the Internet, especially after the huge success of the World Wide Web. However, with the rapid growth of information and easy access of information, a question that may be raised is related on how to find

useful information and knowledge? After the entire semantic web remains a dream concept. The usual advice is to use an Information Retrieval System (IRS) [187].

IRS is a generic name given to a class of tools dedicated to given technologies, such as databases, for the selective manipulation and retrieval of large collections of information in different presentation formats. Typically, an IRS investigates different aspects of the information such as representation, storage, organization, and access. A core assumption of their retrieval/search centric techniques is that their users know exactly what information they seek.

Authors such as Baeza-Yates and Ribeiro-Neto [187] and Yao [188] consider IRS as an extension (or evolution) of the basic search functionality of Data Retrieval Systems (DRS). The reason is simple. Both are focused on the retrieval functionality. However, they also have some differences. Yao [188] asserts that DRS deals with well defined, structured and simple problems while IRS deals with not-so-well defined, semi-structured or unstructured and not simple problems. In other words, DRS typically work with database system while IRS investigates different aspects of the information such as representation, storage, organization, and access. In addition, the core assumption of DRS is that the data items and user information needs can be precisely described while the core assumption of an IRS is that their users know exactly what information they seek.

Despite being the natural evolution of the information retrieval area, coupled with the rapid growth of the Web, IRS began experiencing problems due to their design philosophy and principles. According to Yao [188], these problems are caused by the emphasis on the storage and search functionalities, since an IRS performs search at the raw data level, instead of the model level, and without user interaction. In other words, trying to find useful and relevant information part of a large collection of unstructured documents is undoubtedly a cumbersome task.

In order to solve this problem, Yao and Yao [189][190] proposed to shift the focus of IRS from a system centric to a user centric and from a retrieval centric to a support centric design philosophy. This culminated in what is known as Information Retrieval Support Systems (IRSS). The main goal of any IRSS is to support users, providing the necessary means, tools and languages to facilitate the task of finding useful information to its users and managing it. In other words, IRSS set their focus on the supporting functionalities for the users rather than concerning themselves with the underlying retrieval related functionalities.

In IRSS, the users play more active and important roles. They can, for example, take decisions at various stages of retrieval and find useful information process. With exploratory search and browsing, only users may determine the relevance of each information item. In addition, in some occasions, a user may not want details about particular data, but rather a general view before going to a more in-depth analysis. Unlike IRS that presents search results in the form of ranked list, an IRSS user would be able to use graphical and visualization outputs to view a result, increasing the level of inference and analysis.

However, in the context of Web, the focus on user interests does raise certain issues. This is especially due to the fact that a user may not know exactly what is being searched for inasmuch as billions of Web pages daily updated. For this reason, IRSS applied to Web are known as Web Information Retrieval Support Systems (WIRSS). According to Hoerber [191], WIRSS apply intelligent methods and advanced Web-based technologies over the traditional focus on the automated search within digital

collections, to enable users to better specify their information needs, evaluating and exploring search results, and managing the recovered information.

For example, what is the answer to the question: How many times have Brazil won the FIFA World Cup? Among the results returned there will those reflecting the fact that the Web knows, for instance, that Brazil will host the 2014 World Soccer Cup competition and that the 2010 World Soccer Cup takes place on South Africa, even in sites that do not consider the initial query terms. On the other hand, for a WIRSS with support for semantic search, the answer would be five.

Two examples that illustrate such functionalities, taken from the Web, are Google maps [192] and AllinOneNews [193]. The former is a heterogeneous WIRSS, where the user queries by postal address information and receives topological map data, terrain and geospatial data, high-resolution satellite imagery data, real-time traffic flow stream data from sensors, and pictures of neighborhoods in the Web browser. The latter is a news meta-search engine that integrates homogeneous information sources, where the user queries are dispatched to a selected (based on the query) subset of 10-20 most promising sources from a list of 1800 online news sources. The results are merged and top ranked, before being presented [193].

However, the design and deployment of WIRSS introduce new challenges as precisely pointed by [191][194]. The main one is the evolution of traditional Search Engines (SE), based on classical IRS concepts, to Search Support Engines (SSE) focused on providing different supporting functionalities for end users. Authors such as Zeng *et al.* [194] and Marchionini and White [195] confirm SSE requirements, besides offering typical and traditional navigation, search and browser functions, to additionally implement important user oriented supporting functionalities such as knowledge organization, discovery, and visualization. To prove and evaluate such approach, they developed a layered SSE focused on the management of the DBLP dataset [196], known as DBLP-SSE.

Another challenge is related to the representation of Web search results. The typical list-based representation of results is extremely effective when the information being sought is very specific. However, when the queries provided by the searchers are ill-defined, vague, or ambiguous, the list provides little support for the searcher to discover the few relevant documents from the many irrelevant ones. Tilsner *et al.* [197] implement a prototype to Web search based on fuzzy clustering and visualization, named *CubanSea*, capable to provide a novel method for representing search results, allowing that users have an active role in the search process, making high-level selections of fuzzy clusters of documents, thereby reducing the number of non-relevant documents within the search results list.

Recently, a different approach proposed by Marchionini and White [195] introduced the concept of Information Seeking Support System (ISSS), which emphasizes the necessity of shifting from the study of Information Seeking towards Seeking Support. The authors argued that seeking information for learning, decision making, and similar complex mental activities that take place over repeating time periods, requires the development of new specially designed solutions and support services to help users managing, analyzing, and sharing the built up knowledge.

Current ISSS works cover a wide range of functionalities. Shah [198] developed a framework, named *ContextMiner*, capable of executing automated crawls on various Web sources and collecting data as well as contextual information. The *ContextMiner* may analyze and add value to collected data and its context, and continuously monitors

digital objects of interest to its users over time. WolframAlpha [199] is seen as a famous computational knowledge (semantic) engine able to answer queries directly by computing the result from structured data, rather than providing large numbers of pointers to documents or Web pages as is typically the case with existing search data indexing engines. Relation Browser [200][201] provides a dynamic user interface that allows users to explore a data set through the use of faceted browsing and keyword search. Developed as a Java applet, it focuses the understanding of the relationships between items in a collection and the exploration of information spaces (e.g., a set of documents or Web pages).

7.2. ARAPONGA Overview

The adoption of web search technologies led people to expect immediate and easy access to information in all aspects of life. Web-based search services have become fundamental components of the cyber infrastructure that supports economic growth and social advance. Although there is a built base of solutions for search, human needs for information go beyond search to filtering, assessing, sense-making, synthesizing, and using information to even meeting the drive for the creation of new knowledge.

A special and focused case is that of security information. Although there are a great number of given Web sites built to manage vulnerabilities reports, lists of malicious hosts (typically IP, DNS servers, domains and ASN), graphics and statistics, network administrators and IT managers are forced to painfully search thoroughly to found such information.

To such end, the present work develops and evaluates a tool called ARAPONGA based on the basic principles of Web-based Information Retrieval Support Systems (WIRSS) and Information Seeking Support Systems (ISSS). It is capable to provide: (i) integration of Web crawled content into a single place; (ii) a powerful search support engine focused on security; (iii) a unified and simple access with support for logical expressions; and (iv) interfaces to deal with both human users as external system like search engines and decision making tools.

ARAPONGA provides automated searches content on vulnerabilities and malicious activity statistics published on the Web, storing them, and, finally, allowing fast and easy access to this information directly for both users and other systems. More specifically, ARAPONGA offers two types of supporting functionalities: search refinement support and knowledge analysis support.

Search refinement support describes a set of searches that can be performed by users and external system alike to found security information content. This functionality can be compared with the traditional and advanced searches available on the most vary Web search engines of the Internet. Although ARAPONGA also provides typical search features as query input text box and search results listing, it also distinguishes itself. ARAPONGA employs the concept of templates at indexing Web pages aiming to extract better and more specific contents. Consequently, search results become more consistent and relevant with users' interests since there is a major diversity of complements to help the search such as specific tags, kind of pages and knowledge domain. Templates will be explained in next section.

As far as is concerned to search refinement support functionality, ARAPONGA is able to answer some specific questions as the following:

- Is my IP address, server, domain or ASN related to some type of suspicious or malicious activity, intrusion or attack reported on the Internet?
- Is there some new type of anomaly (attack, intrusion or similar) related to a specific type of service, port or protocol?
- What is the last appearance of a particular anomaly or vulnerability?
- What are the vulnerabilities related to a specific product (software or hardware) or given vendor?

The second main ARAPONGA functionality is knowledge analysis support. It allows the elaboration (building) of various types of structures (knowledge and statistical graphics) to represent a specific knowledge domain. For example, ARAPONGA can model and present the frequency distribution of vulnerabilities or knowledge domains. Figure 7.1 illustrates a knowledge structure of the “vendor” search term, which products are related with vulnerabilities.

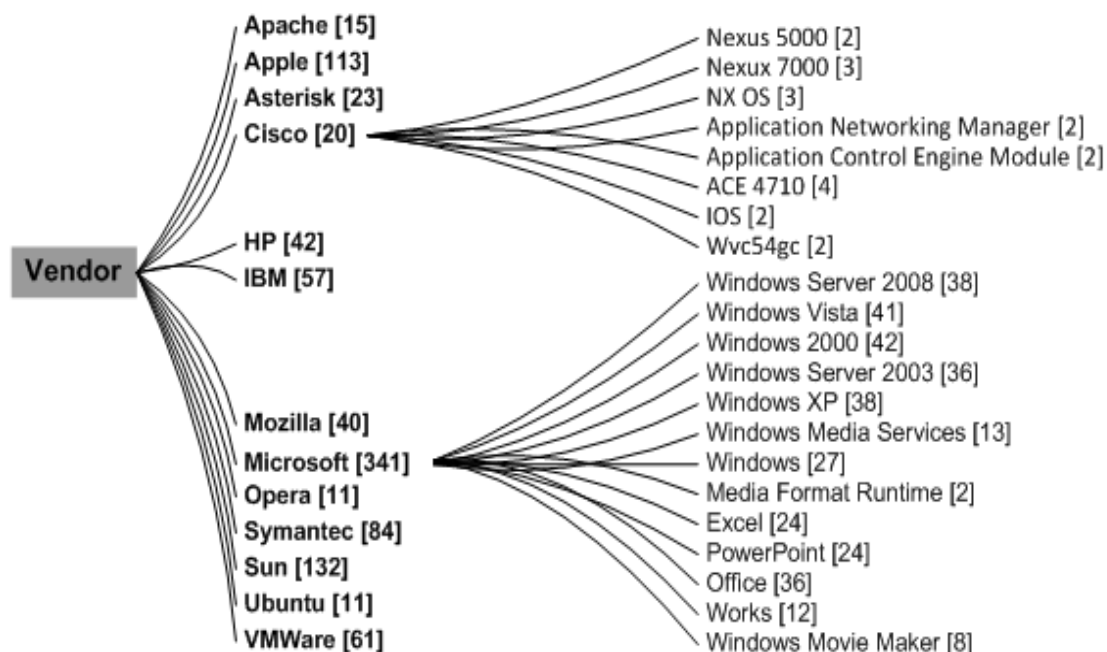


FIGURE 7.1: A partial multi-level knowledge structure for vulnerability by vendor

Based on all indexed documents, ARAPONGA may infer that if one needs very general information with respect to “vulnerability” by vendor, the seeker may just want the knowledge in the second level (the first level just has one node “vulnerability by vendor”). Furthermore, if the user requires more detailed knowledge concerning one branch of vendor, it can choose, for example, “Microsoft” and get a finer grained structure. This way, it is possible to produce a scalable knowledge structure which provides the knowledge source in different levels of details with an interactive manner concerning different user needs.

It is important emphasize that knowledge analysis support functionality is mainly user centered since the system generates relevant analysis results to help these understand the security threats for a given context.

7.2.1. Templates

Typically, Web crawling technologies assume content extraction directly from Web pages without relying on the actual keywords declared in the HTML source code. The key reason is that Web publishers usually define keywords different from the actual content. Consequently, indices generated by indexing process are common or generic since they are based on the main subject of the pages.

Despite the adoption of RDF (Resource Description Framework) [202] as standard model for data interchange on the Web, the problems involving index generation continued because there is no guarantee for Web crawlers that the information included in a RDF file fully corresponds to actual content. Moreover, RDF description prepared for a specific Web page provides information about the content and does not include any information about content allocation on that same Web page.

In order to solve this problem and also improve the results of the indexing process, ARAPONGA employs the concept of templates. The term *template* refers to a model to represent some content, while providing a kind of standard of visualization. ARAPONGA templates have keywords (tags, fields or regions) often found in relevant Web pages that follow some predefined standard. Templates can be created to represent a complete Web site or domain inasmuch as the contents and structures are oftentimes repeated. However, every time a Web site or domain introduces new different structures, several templates need to be re-generated to represent such differences.

In ARAPONGA, templates try to establish a relationship between URLs and keywords. This way, after the index process, there is not a unique search space, but a multitude of search spaces are maintained for diversity, increasing the probability of a particular topic being relevant. Figure 7.2 exemplifies the difference between a normal search space and a search space with templates.

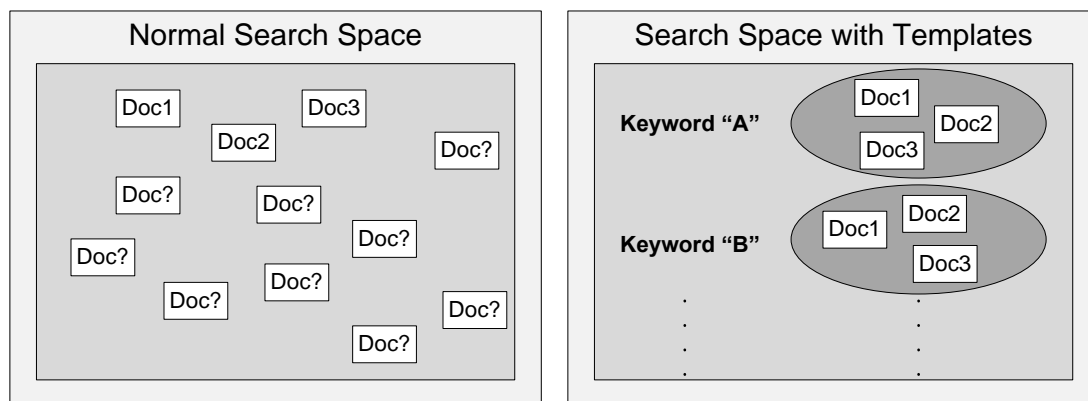


FIGURE 7.2: Differences among search spaces

In order to clarify the idea behind ARAPONGA templates, Figure 7.3 shows an example for Secunia's Advisories Web site [203], a famous and reliable repository about vulnerabilities, where the black rectangles represent the fields identified as useful whereas the red round ones represent the content of each field. Table 7.1 describes each field and its contents.

Community

Advisories

- Database
- Search
- Advisories by Product
- Advisories by Vendor
- Terminology
- Report vulnerability

Research

Forum

My Profile

Our Commitment

Community Login

Username:

Password:

Login

[Register now](#)
[Forgot password?](#)

Secunia Advisory SA38860

Internet Explorer iepeers.dll Use-After-Free Vulnerability

Secunia Advisory SA38860

Release Date 2010-03-09

Last Update 2010-03-15

Popularity 11,735 views

Comments 6 comments

Criticality level Extremely critical

Impact System access

Where From remote

Authentication level Available in Customer Area

Report reliability Available in Customer Area

Solution Status Unpatched

Secunia PoC Available in Customer Area

Secunia analysis Available in Customer Area

Systems affected Available in Customer Area

Approve distribution Available in Customer Area

Software: Microsoft Internet Explorer 6.x, Microsoft Internet Explorer 7.x

Secunia CVSS Score Available in Customer Area

CVE Reference(s) CVE-2010-0806 CVSS available in Customer Area

Description

A vulnerability has been discovered in Internet Explorer, which can be exploited by malicious people to compromise a user's system.

The vulnerability is caused due to a use-after-free error in iepeers.dll when handling invalid values passed to the "setAttribute()" function. This can be exploited to dereference invalid memory when a specially crafted web page using the "#default#userData" behavior is accessed.

Successful exploitation allows execution of arbitrary code.

NOTE: The vulnerability is currently being actively exploited.

Solution

Set the Internet zone security setting to "High" or disable Active Scripting support.
Further details available in Customer Area

Provided and/or discovered by

Reported as a 0-day.

Changelog

Further details available in Customer Area

Track and eliminate the complete Vulnerability threat lifecycle

GET ACCESS

Track critical vulnerabilities affecting your infrastructure instantly

GET ACCESS

Latest advisories

New advisories: 17
New vulnerabilities: 26
Updated advisories: 29

- 65 views [Open Web Analytics "IP" File Inclusion Vulnerability](#) **New**
- 99 views [TSOKA CMS "id" SQL Injection Vulnerability](#) **New**
- 87 views [SiteX CMS Local File Inclusion and SQL Injection Vulnerabilities](#) **New**
- 116 views [Post Card "catid" SQL Injection Vulnerability](#) **New**
- 108 views [Deliver File Handling Multiple Security Issues](#) **New**
- 128 views [IBM WEBI Unspecified Cross-Site Scripting Vulnerabilities](#)
- 103 views [CF Image Hosting Script "img" File Disclosure Vulnerability](#)
- 107 views [N-13 News "default_login_language" Local File Inclusion Vulnerability](#)
- 127 views [Fedora update for krb5](#)

[See all](#)

FIGURE 7.3: Secunia Web site Template

TABLE 7.1: Secunia Advisories template

Template Fields	Description
<i>Secunia Advisory</i>	Defines the unique identifier of the vulnerability by Secunia standard
<i>Release date</i>	Indicates the release date of the vulnerability
<i>Last update</i>	Indicates the last update of the vulnerability
<i>Popularity</i>	Indicates the popularity of the vulnerability according to the number of access of the Web page
<i>Critical</i>	Exposes the level of severity of the vulnerability
<i>Impact</i>	Describes how the vulnerability affect the targets
<i>Where</i>	Describes where the vulnerability is achieved
<i>Solution Status</i>	Describes what was done to solve the problem
<i>Software</i>	Describes the software or systems affected by vulnerability
<i>CVE-Reference</i>	Indicates the CVE identification of the vulnerability
<i>Description</i>	Describes in general lines the vulnerability
<i>Solution</i>	Describes how to solve the vulnerability

The use of templates offers the possibility to identify more specific content and indexing this as keyword. In comparison with a common index process, where the generated search space is composed by all content, the present indexing process using templates generates a more reduced search space composed by content indexed using keywords.

As result, the use of templates allows a performance gain in terms of response time and processing. In ARAPONGA, this type of search is called an advanced search. It is important emphasize that the search process for both simple and advanced search remains the same (regarding searching, sorting and page ranking). What really changes is the search space and the number of returned pages. A simple search is likely to return much more results than the advanced one.

7.2.2. Architecture

The ARAPONGA architecture is modular, to ease future modification, and the seamless addition of new components. As shown in Figure 7.4, the architectural components were broken down in accordance with their role.

The entities that compose the ARAPONGA system are:

- **Crawler** – responsible for harvesting Web pages and associated metadata and contextual information, where the sources of interest are defined in an initial URL list (seeds). When a URL is accessed, the crawler downloads the page associated to it and analyzes all links it finds there in order to select and store these links into a list of new URLs to visit. This process is repeated until some stopping condition is achieved.
- **Indexer** – responsible for indexing the collected content by the crawler. However, instead of the simple indexing based on the traditional content of pages (content, URL, timestamp and other identifiers), this component makes use of known templates to extract different and more specific field identifiers and, consequently, permits highly refined searches. The indexer is also responsible for the removal of Web pages that are indexed if they lack relevant content to the security context.
- **Search Engine** – receives queries and returns the answers in an orderly manner. This component performs two distinct tasks: searching and ranking. The search module takes in queries in a natural language like form interface component. These are then translated into accepted queries and the base is looked up for documents. The results are then passed to the next module. The ranking module quantifies the relevance of received documents with regard to the initial query and returns these to the interface component.
- **Interface** – offers a front-end between users (human operators and external systems) and ARAPONGA (search engine component, more specifically), providing different types of input and output interfaces for execution of queries and display of responses. This component performs two main functions: query handler and output viewing. The former is responsible to translate originals queries for the format accepted by the search engine component. The latter displays the queries' results (typically a Web page, but also uses graphics and ordered XML list).

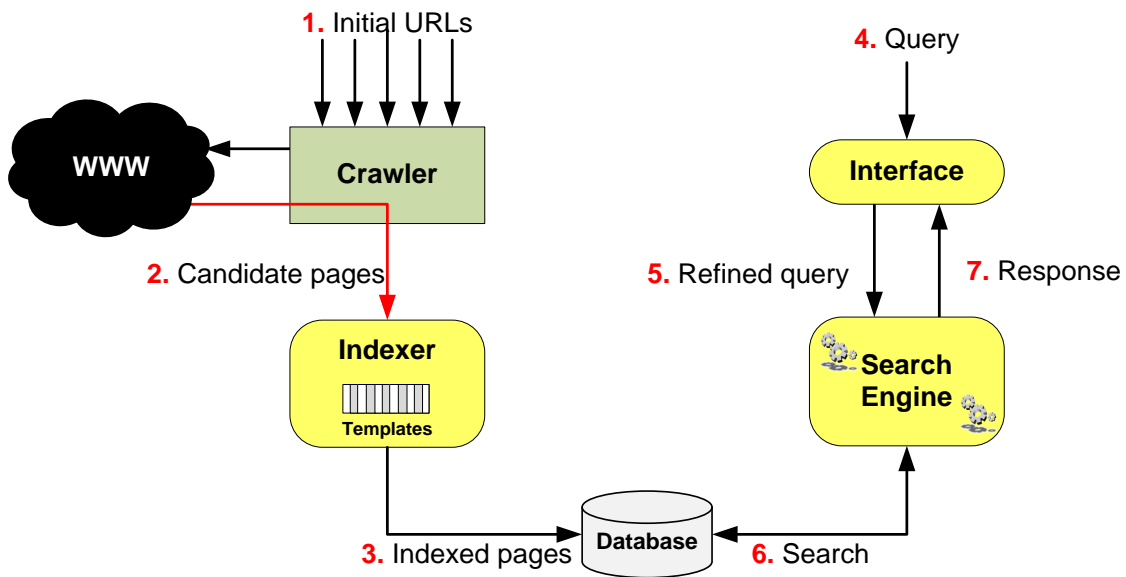


FIGURE 7.4: Architectural and workflow diagram of ARAPONGA

7.2.3. Components Interaction

The interaction between ARAPONGA components can be divided in two stages. First, there is the processing of collecting and indexing Web pages. It involves the crawler and indexer components (detailed on Steps 1, 2 and 3 in Figure 7.4). In Step 1, the crawler is fed with initial URLs (or domains) that had been listed as information source for vulnerabilities and Internet anomalies. Next, in Step 2, the crawled Web sites (candidate pages) are saved and the indexer component starts using the templates to index all these content. As a result, in Step 3, the relevant and useful files are saved in the indexed files base.

The second stage involves querying (Steps 4, 5, 6, and 7). To help the reader gain familiarity with the ARAPONGA system, an example is used. Supposing that an IT manager desires to know if: there are TCP alerts on port 80 related to attacks, bots and spam in the period ranging from 01/01/2010 and 01/07/2010?

Upon receiving this query, the interface component translates it and formulates a new one according to predefined parameters. If some part of question is not understood, that is, cannot be translated, the question is discarded and an error message is displayed to the user. Otherwise, in this example, the following is generated:

TCP 80 *-pageType* “**attack,botnet,spam**” *-date* “01/01/2010 to 01/07/2010”

Upon its receipt by the search engine component, the query goes through its own validation process, where insignificant or unimportant words are removed using an efficient stop-words removal technique. Next, the search engine initiates the search from the query on Web pages stored in its local database. The goal is to verify which pages contain the information relevant to the present query (in this example TCP 80) according to the specified parameters (*-pageType* “attack, botnet, spam” and *date* “01/01/2010-01/07/2010”). The pages that fit this profile (respond positively to these rules) are ranked and listed in an ordered in a decreasing way according to the ranking field. Finally, the search engine returns the query result to the interface component.

7.3. Implementation

This section describes the implementation process of ARAPONGA while specifically focusing on its components and their integration.

However, before describing ARAPONGA development, two important design aspects in this project need to be clarified: content selection and prerequisites supported by the crawler tool.

7.3.1. Preliminary Questions

Content Selection

Content selection is decisively important as it defines what type of Web sites are more suitable for delivering relevant and updated information on Internet traffic anomalies and vulnerability reports. There are indeed scores of Web sites offering security and traffic anomaly reports. Some of these have an open access policy whereas others require some level of subscription and hence have an extra obstacle for crawlers.

The question is: where to get reliable security information? The answer can be found in a great diversity of sources such as ATLAS [63], ShadowServer [64], Secunia [203], NVD [204], US-CERT [205], KB-CERT [206], OSVDB [207], ISS [208], DragonSoft [209], SecurityFocus [210], SenderBase [211], ThreatExpert [212] and Team Cymru [213].

In order to answer this question, two metrics to evaluate the general and relevant characteristics of this kind of Web sites are adopted. These metrics better portray some significant particularity of the content as well as the some features or aspects important for the crawler process. Similarly to other evaluation works, metrics in the form of simple discrete or continuous values to describe features such as the amount of recorded information and the update time. These metrics are the following:

- **Update time** - indicates the frequency of the information updating. This metric is important because it allows measuring the time interval in which a search engine should visit the pages of a certain area.
- **Access content** - indicates if the information is accessible with or without the need of authentication and if there is some type of restriction to access the content. These metrics permit to identify the information sources where the access is easier.

After analyzing the sources according to the established selection criterion, the following Web sites have been chosen for ARAPONGA access:

- Secunia [203], NVD [204], US-CERT [205] KB-CERT [206], DragonSoft [209] and SecurityFocus [210] for records and reports of vulnerabilities, and;
- ATLAS [63], ShadowServer [64], ThreatExpert [212], and Team Cymru [213] for malware and Internet anomaly statistics.

Table 7.2 summarizes the results of the comparisons among all previously listed using the defined metrics.

TABLE 7.2: Comparison among various Web sites

	Update time	Access content
<i>ATLAS</i>	Daily	Part of the content needs of authentication
<i>DragonSoft</i>	Daily	No
<i>ISS</i>	Unknown	No
<i>KB-CERT</i>	Daily	No
<i>NVD</i>	Daily	No
<i>OSVDB</i>	Daily	No
<i>Secunia</i>	Daily	No
<i>SecurityFocus</i>	Daily	No
<i>SenderBase</i>	Daily	Meta-tag limitation for crawlers
<i>ShadowServer</i>	Daily	No
<i>Team Cymru</i>	2 hours	No
<i>ThreatExpert</i>	Daily	No
<i>US-CERT</i>	Daily	No

The other Web sites could not be selected due to some issues. As far as is concerned vulnerabilities Web sites, OSVDB [207] can be considered among the best representatives due to its huge amount of recorded information, without the necessity of authentication, and access support of all content by different ways. However, it fails with regard to the completeness of its information. The work of Borba [214] shows that, as of June 05 2009, the OSVDB base had 54.004 records, where 41.597 records present some kind of unknown or simple null information. This problem was observed in both old and new records, which confirms a serious problem of completeness. ISS Web site [208] was also discarded due to the fact that its update time is unknown. There are times were more than two weeks passed without any change.

Regarding malware and Internet statistic Web sites, SenderBase [211] was not chosen since its information is only available in the form of graphs or figures. The small amount of content gathered is not representative.

Crawler tool

The second design issue has to do with which crawler to use. After some studies and exchanges with experts in Web mining information, it was decided to evaluate WIRE [215][216], an open-source crawler. The choice by this tool was based on three features:

- **Scalability** – WIRE was designed to work with large volumes of documents, and was positively tested with several million documents. The current implementation would require further work to scale to billions of documents (e.g.: process some data structures on disk instead of in memory).
- **Configuration** - all its parameters for crawling and indexing can be configured, including several scheduling policies.
- **Performance** – WIRE is entirely written in C/C++ for high performance. The downloader modules of the WIRE crawler (“harvesters”) can be executed over several machines in parallel.

However, after initial tests utilizing the previous selected content, the WIRE crawler revealed some issues with regards to ARAPONGA requirements. The main one was the lack of authentication support. For example, one of the chosen Web sites requires authentication when asked for further information on specific IP addresses involved in DDoS attacks. Although WIRE is open source, the deployment of a new module to support this requirement is difficult due to the limited available documentation and unclear code. For these reason, the use of WIRE as ARAPONGA crawler was abandoned.

After further consultations and studies, two crawlers were tested: Heritrix [217] and Nutch [218]. Although both are developed in Java, they are very used both in academic as commercial environments.

Developed by Internet Archive, Heritrix [217] is an open source and Web crawler. Heritrix is a generic crawling framework into which various interchangeable components can be plugged, enabling diverse collection and archival strategies, and supporting the incremental evolution of the crawler. Totally developed in Java, it employs text files to configure aspects such as initial URLs, environments, and Java options. In addition, Heritrix fetches a vast variety of contents.

Nutch [218] is an open source Web-search tool capable of executing Web-specific functions such as crawling and parsing HTML and other document formats. Developed by the Apache Foundation in Java, Nutch utilizes the Lucene library [219] to index the contents. Nutch has also been widely used to implement a fast and efficient mechanism for searching and indexing documents in several formats. Nutch employs XML files to configure aspects such as initial URLs, operation, and plugins (new or not). Although it fetches contents on different formats it not recognizes images.

Since Heritrix and Nutch are very similar, both were evaluate in terms of features. Table 7.3 summarizes the comparisons among these crawler tools.

TABLE 7.3: Heritrix and Nutch comparison

Features	Heritrix	Nutch
Source Code Changes	The source code is documented, but there is little material to help a user to develop or maintain the Heritrix source code.	All source code is very well documented. In addition, there are many examples and tutorials to create and handle plugins.
MapReduce²⁷	Available through an extension.	Implemented as part of Nutch.
Authentication	Offers two types: Basic and Digest authentication and POST and GET of an HTML Form.	Offer HTTPS, NTLM (NT LAN Manager), Basic and Digest authentication schemes.
Clustering	Utilizes Heritrix Cluster Controller (HCC), a set of packages that enable control of a cluster of Heritrix instances running across multiple machines.	Available as a plugin (Carriot Project [220]) included in Nutch codebase.
Plugins	Provide several types of pluggable modules.	Provide a large part of the functionality of Nutch. Writing or

²⁷ MapReduce is a programming model and an associated implementation for processing and generating large data sets.

		handling a Nutch plugin is easy.
Parallel operation and Distributed File System	Employs Hadoop ²⁸ Distributed File System (HDFS) to stores large files (multiples of 64 MB), across multiple machines. It achieves reliability by replicating the data across multiple hosts, and hence does not require RAID storage on hosts.	Employs Nutch Distributed File System (NDFS), a set of software for storing very large stream-oriented files over a set of commodity computers. These two facilities are provided by Hadoop Project [221].
Parallel operation	Supports multiple fetches and distributed search through diverse plugins.	Supports multiple simultaneous fetches; parallel and distributed db update; and distributed search.

Based on the features presented in Table 7.3 and considering installation and operation tests realized, the choice ultimately fell on the Nutch crawler.

7.3.2. Components

Crawler

As previously mentioned, Nutch plays the role of a crawler component within ARAPONGA. In order to deal with issues related to the quantity and quality of crawled information, Nutch makes use of limiters and filters. For example, since the Web has a huge number of pages and the rapid update of content increases the likelihood of outdated content being downloaded, it employs depth limiters, to avoid huge deviations from the top level, and amplitude limiters, to restrict the number of links in pages that can be referenced. In addition, URL filters are also used to delimit specific URL for consultation.

Indexer

To implement the indexer of the collected content, Java (version 1.6), Lucene API [219] and Jericho HTML Parser [222] are used.

- Lucene [219] is a high-performance search engine library written entirely in Java and open source. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform software. In addition, Lucene offers through a simple API efficient search algorithms, including field oriented searching, many powerful query types, multiple index searching, and simultaneous update and searching operations. Furthermore, the Lucene ranking function, the core of any search engine applied to determine how relevant a document is to a given query, is built on a combination of the Vector Space Model (VSM) and the Boolean model of Information Retrieval. Lucene uses also the Boolean model to first narrow down the documents that need to be scored based on the use of Boolean logic in the query specification
- Jericho HTML Parser [222] is an open source java library for the analysis and high-level manipulation of parts of an HTML document, including

²⁸ Hadoop Map/Reduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner [221].

server-side tags, while reproducing verbatim any unrecognized or invalid HTML. It also provides high-level HTML form manipulation functions.

Table 5.4 presents other features of Lucene and Jericho.

TABLE 7.4: Features of Lucene and Jericho

Lucene	Jericho
Ranked searching: best results returned first	The presence of badly formatted HTML does not interfere with the parsing of the rest of the document.
Many powerful query types: phrase queries, wildcard queries, proximity queries, exact phrase queries, range queries for date/time and number values.	ASP, JSP, PSP, PHP and Mason server tags are explicitly recognized by the parser.
Fielded searching: all fields are searchable as a whole or each field separately.	Compared to a tree based parser such as DOM, the memory and resource requirements can be far better if only small sections of the document need to be parsed or modified.
Boolean operators: any combination between search terms (AND, OR, NOT).	Compared to an event based parser such as SAX, the interface is on a much higher level and more intuitive, and a tree representation of the document element hierarchy is easily created if required
Multiple index searching with merged results.	Custom tag types can be easily defined and registered for recognition by the parser.
Simultaneous searching and updates.	Provides a simple but comprehensive interface for the analysis and manipulation of HTML form controls, including the extraction and population of initial values, and conversion to read-only or data display modes
	Analysis of the form controls also allows data received from the form to be stored and presented in an appropriate manner.

Back to the Indexed component (or Lucene), its operation can basically be divided in three steps:

1. The Lucene API is used to acquire all downloaded Web pages.
2. For each received page, the Jericho library is used to make a comparison between page titles and a set of predefined identifiers, aiming to identify templates that help reference this page. Once a template for this page is matched, the page has its context extracted and identified and each information block (part of content) has an associated keyword. On the other hand, if no template is found as a match for a given page, a generic common

indexing is made for such page as explained in section 7.3.3 (Interaction between components).

3. Lastly, Lucene adds the timestamp, title, and URL identifiers, to the internal control system identifiers.

After executing these steps, Lucene is invoked to index the page.

Search Engine

Similarly to the other ARAPONGA components, the search engine is implemented in Java. It is responsible for performing search and inference tasks and results analysis before their presentation at the interface component.

To perform such activity, ARAPONGA's search engine makes use of the Lucene API. Basically, upon receiving a query, it starts a comparison process with the documents already available in the local database. Next, the selected documents are ranked, sorted in descending order, and sent to interface component.

Interface

The last, but not least, ARAPONGA component is its interface. It intermediates the communication between users (both human and external systems) and the search engine, providing different types of input and output interfaces for execution of queries and the display of responses.

To achieve such functionality, many types of queries are supported. Table 7.5 describes and exemplifies each one of them.

TABLE 7.5: Query types of ARAPONGA Interface

Query Type	Description
Simple	<p>Represent a common and generic query composed by a unique input, the term or terms of interest. In addition, this query also allows searches among time periods using the parameter <i>date</i>.</p> <p>The search is executed over all indexed content. As result, this query returns an ordered list.</p> <p>Examples of this query are: “<i>botnet</i>” or “<i>Internet Explorer -date 01/01/2009 to 12/31/2009</i>”.</p>
Advanced	<p>Represent a detailed query, resulting of the use of templates. This query permits the union of different parameters (inputs), aiming to restrain the search space and provide a more correct answer. It works receiving the term or terms to be searched as input follow by the advanced parameters.</p> <p>Basically, there are three (3) advanced parameters:</p> <ul style="list-style-type: none"> • <i>field</i> – indicates that the searched term(s) must be obligatorily founded along with the specified keyword(s). In the example (<i>AS3462 -field ASN</i>), the parameter <i>-field ASN</i> indicates that the term <i>AS3462</i> only will be searched in Web pages where the keyword <i>ASN</i> is present. • <i>pageType</i> – indicates that the searched term(s) must be obligatorily founded in Web pages which type match with the specified as

input. The example (*Storm Worm -pageType alerts,bulletin*) represents the necessity to search the term *Storm Worm* in Web pages classified as *alerts* and *bulletin*.

- *domain* – indicates that the searched term(s) must be obligatorily founded in Web pages which Web domain match with the specified as input. In the example (*Microsoft -domain secunia.com,cert.org*), the term *Microsoft* only will be searched in Web pages that belongs to the domains specified as input.

In addition, this query also allows searches among time periods using the parameter *date*.

The result of this advance query is an ordered list containing the contents where the term(s) are found. Note that the keywords are extracted from each page during the indexing process made according to specific templates.

Other examples of advanced search are presented as following:

- *SQLInjection -pageType alert,bulleting,database -date 01/01/2010 to 01/31/2010*
- *ASN Brazil -field ASN -pageType report -domain atlas.arbor.net -date 10/20/2009 to 12/31/2009*
- *Microsoft Internet Explorer -field system_affected,software -date 01/20/2010 to 02/10/2010*
- *AS3462 -domain atlas.arbor.net,teamcymru.com -date 02/01/2010 to 02/08/2010*

Malicious

Represent a refined query, as advanced search, resulting of the use of templates. The goal of this query is to identify (confirm) if a determined input is related to a malicious activity. It works by receiving two inputs: the term(s) to be searched and the parameter *-malicious*. Typically, the search term represents an IP address, DNS server, domain or ASN while the parameter *-malicious* is followed by a list of keywords (attacks, spam, phishing, botnet, and so on) that must be searched.

The result of this query is also an ordered list. However, the list is ordered alphabetically according to the malicious activity founded. This query is very useful to investigate specific situations such, for example, if an SMTP server is listed in blacklist or whitelist Web sites.

An example of this query is the following: *gprt.ufpe.br -malicious spam,fastflux,CC,attack*.

Beyond these queries, the interface component offers another distinct query. **Vulnerability summary** is a query intended to generate a list for a specific vulnerability. It takes three inputs: the term(s) (vulnerability, in this case) to be searched, the parameter *-summary_vulnerability*, and the time period parameter *-date*.

This query goes through all contents looking for the searched term(s), returning an XML file containing a description and the following information: (i) the level of criticality or severity of the vulnerability, (ii) the number of times the vulnerability achieved this level, and (iii) where and how to explore the vulnerability.

Figure 7.5 shows the returned result from the following query: *Microsoft – summary_vulnerability -date 09/01/2009 to 04/30/2010*

```

<Summary_Vulnerability>
  <Vulnerability id="1">
    <Title name="Windows SMB2 Remote Denial of Service Test">
    <Description>
      Microsoft Windows Server 2008, Vista are exist array index error in the SMB2 protocol implementation
      in srv2.sys,which could allow remote attacker to cause a denial of service (system crash).
    </Description>
    <Timeline>
      <Disclosure_Date>2009-09-17</Disclosure_Date>
      <Published_Date>2009-09-17</Published_Date>
      <Last_Update>2009-10-13</Last_Update>
      <Solution_Date>2009-10-13</Solution_Date>
    </Timeline>
    <Classification>
      <Attack_From>Remote</Attack_From>
      <Impact risk="high">Denial of Service</Impact>
      <CVSS>7.8</CVSS>
      <CVE_ID>CVE-2009-3103</CVE_ID>
    </Classification>
    <Affected>
      <product>Windows Vista</product>
      <product>Windows Server 2008</product>
    </Affected>
    <Solution>
      Install the patch to fix the problem.
    </Solution>
    <References>
      <name>Microsoft Security Bulletin MS09-050</name>
      <url>http://www.microsoft.com/technet/security/bulletin/ms09-050.msp</url>
      <name>Microsoft Security Advisory (975497)</name>
      <url>http://www.microsoft.com/technet/security/advisory/975497.msp</url>
      <name>Microsoft Fix it for 975497</name>
      <url>http://support.microsoft.com/kb/975497</url>
    </References>
  </Vulnerability>
  <Vulnerability id="2">
    <Title name="Microsoft IE Unspecified Uninitialized Memory Corruption">
    <Description>
      Microsoft Internet Explorer 6, 6 SP1, and 7 does not properly handle objects in memory, which allows
      remote attackers to execute arbitrary code by accessing an object that (1) was not properly initialized
      or (2) is deleted, leading to memory corruption, aka "Uninitialized Memory Corruption Vulnerability.
    </Description>
    ...
    <Classification>
      <Attack_From>Remote</Attack_From>
      <Impact risk="high">Loss of Integrity</Impact>
      <CVSS>9.3</CVSS>
      <CVE_ID>CVE-2010-0267</CVE_ID>
    </Classification>
    ...
  </Vulnerability>
  ...
</Summary_Vulnerability>

```

FIGURE 7.5: Example of Vulnerability summary query by Microsoft term

In practical terms, the interface component offers two types of interfaces. The first one is visual, Web-based, and is intended for human operators (Figure 7.6). The second is a command line-based application and it allows ARAPONGA to interact with external systems.



FIGURE 7.6: ARAPONGA's GUI Interface

7.4. Evaluation and Initial Results

This section presents the evaluation and initial results of the ARAPONGA system.

7.4.1. Performance Metrics

In order to analyze the knowledge obtained by ARAPONGA, the performance evaluation metrics, initially proposed by Cleverdon [223] and very used in information retrieval area, based on relevance concept are adopted. In other words, a document is considered relevant when it has importance for the queried topic.

The metrics are:

- **Precision** – defined as the fraction of documents retrieved that are relevant in relation to all retrieved documents.

$$\text{Precision} = \frac{N_{\text{retrieved}} \cap N_{\text{relevant}}}{N_{\text{retrieved}}}, \quad (1)$$

- **Recall** – defined as the fraction of the documents that are relevant to the query that are successfully retrieved.

$$\text{Recall} = \frac{N_{\text{retrieved}} \cap N_{\text{relevant}}}{N_{\text{relevant}}}, \quad (2)$$

In short, precision is the percentage of recovered items that are relevant. Recall is the percentage of relevant items that were recovered. For example, a query with a value of accuracy equal to 0.70 means that 70 percent of the recovered items are relevant, while a query with recall value equal to 0.70 has only 70 percent of the documents are or could be relevant.

7.4.2. Evaluation Methodology

All stages of development, evaluation and tests of the ARAPONGA system were performed in the GPRT²⁹ laboratory (Network and Telecommunication Research Group) of the Federal University of Pernambuco (UFPE). The environment was composed by a PC (Intel Core2Duo T5300, 2 GB of RAM, and 250 GB of HDD, 10/100 Mbps Ethernet interface), running Ubuntu 8.4 Linux distribution. The laboratory keeps a sustained rate of 1 Gbps with PoP-PE (RNP³⁰ point of presence of the State of Pernambuco).

Considering that the objective is to evaluate the performance and features of ARAPONGA, the base of collected content was divided in two. The first one was used to validate and ensure that the implementation was in accordance to the specifications. It was conducted in January 2010, from 4th to 29th, where *Crawler* component of ARAPONGA was set to capture 10 references (links) per page and with depth of 2 references using only ATLAS [63], Secunia [203] and US-CERT [205] Web sites as information source.

To extract visible results, all collected Web pages were indexed during this period using both ARAPONGA and Lucene. The goal is to establish a comparison between them and thus to prove the relevance and efficiency of ARAPONGA's templates. Table 7.6 shows the index results of this experiment.

TABLE 7.6: Base evaluation

Day	Visited URL	Pages indexed with Lucene	ARAPONGA		
			Pages indexed with template	Pages indexed without template	Total of pages indexed
4	2708	347	87	100	187
7	2814	364	92	84	176
8	4417	359	83	78	161
9	3007	334	91	105	196
13	4367	346	79	75	154
15	2855	344	88	100	188
16	2887	351	85	84	169
17	2932	357	89	86	175
19	2888	347	86	115	201
20	2799	343	81	95	176
21	2943	368	84	82	166
22	2800	363	83	95	178
23	2974	354	107	109	216

²⁹ <http://www.gprt.ufpe.br>

³⁰ <http://www.rnp.br>

24	2904	350	85	78	163
26	4416	360	84	80	164
27	2779	339	85	93	178
28	2820	355	93	99	192
29	2732	346	86	87	173
Total	57570	6327			3213

The analysis of the table results reveals a striking difference between pages indexed by Lucene, from now on referred to as the general base, and the pages indexed by the ARAPONGA system. While the general base built 6327 pages, ARAPONGA managed 3213 pages (non indexed pages are not considered). Altogether, such difference is attributed to the use of templates, which permits a more detailed information extraction.

Figure 7.7 clearly shows the difference among pages indexed by Lucene and ARAPONGA.

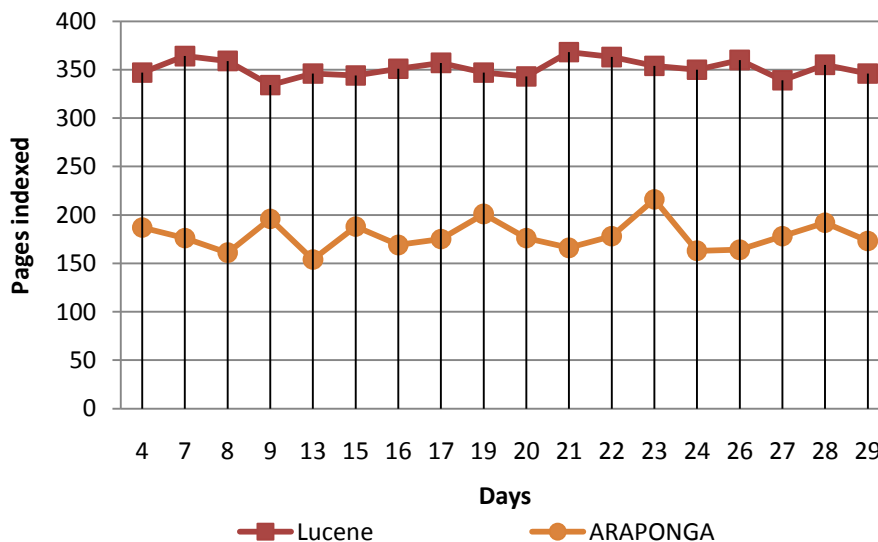


FIGURE 7.7: Comparison ARAPONGA's GUI Interface

Note that there were configuration and operation failures with ARAPONGA's *Crawler* tool (Nutch) in the days 5, 6, 10 and 11. In addition, the days 12, 14, 18 and 25 have no data due to scheduled power shutdowns.

The second base was used to evaluate the support functionalities offered by ARAPONGA such as advanced searches and knowledge analysis through the use of graphics outputs. The evaluated information at this stage is composed by all collected content from the period of February 1, 2010 to March 31, 2010. The *Crawler* component of ARAPONGA was set to capture 150 references (links) per page and with depth of 20 references using all information source defined in Section 7.3.1.

7.4.3. Experiment Results

Performance

In order to evaluate the ARAPONGA's performance, a simple query composed by three terms (Microsoft, vulnerability and high severity) was executed. The idea is to find vulnerabilities considered to have high level of severity involving Microsoft products.

Table 7.7 provides a comparative study of search results from the query performed on the first base, indexed both Lucene (general) as ARAPONGA.

TABLE 7.7: Comparative study of search results from the query

Query	<i>Microsoft + vulnerability + high severity</i>
Results of general base	Number of returned URLs: 3309 List of main URLs: <ol style="list-style-type: none"> 1. US-CERT Technical Alert - Microsoft Updates for Multiple Vulnerabilities (136 pages) 2. US-CERT Security Alert - Microsoft Updates for Multiple Vulnerabilities (408 pages) 3. Secunia Advisories - Vulnerability Information (747 pages) 4. ATLAS Report (Global, Service, Summary, and Vulnerability) (349 pages) 5. US-CERT Cyber Security Bulletin (826 pages) 6. Other (843 pages)
Results of ARAPONGA base	Number of returned URLs: 25 List of main URLs: <ol style="list-style-type: none"> 1. US-CERT Cyber Security Bulletin SB10-025 2. US-CERT Cyber Security Bulletin SB10-018 3. US-CERT Cyber Security Bulletin SB09-355 4. US-CERT Cyber Security Bulletin SB09-348 5. US-CERT Cyber Security Bulletin SB09-327 6. US-CERT Cyber Security Bulletin SB09-258 ...

Quantitatively, the query applied into the general base returned 3309 pages containing references for Microsoft, where only 25 pages described vulnerabilities with high level of severity. Thus, the precision in this base was 0.75%, i.e., only 25 pages were relevant from a total of 3309 documents retrieved.

$$\text{Precision} = \frac{N_{\text{retrieved}} \cap N_{\text{relevant}}}{N_{\text{retrieved}}} = \frac{3309 \cap 25}{3309} = \frac{25}{3309} = 0,0075$$

The recall was 0.39% due to the fact that only 25 pages were relevant from a total of 6327 documents.

$$\text{Recall} = \frac{N_{\text{retrieved}} \cap N_{\text{relevant}}}{N_{\text{relevant}}} = \frac{3309 \cap 25}{6327} = \frac{25}{6327} = 0,0039$$

Regarding the refined base, the query also returned 25 pages containing references for Microsoft, where all of them described vulnerabilities with high level of severity. Thus, the achieved precision in the refined base was 100%. The recall was 0.77%, since 25 relevant pages were retrieved from 3213 available ones.

$$\text{Precision} = \frac{N_{\text{retrieved}} \cap N_{\text{relevant}}}{N_{\text{retrieved}}} = \frac{25 \cap 25}{25} = \frac{25}{25} = 1$$

$$\text{Recall} = \frac{N_{\text{retrieved}} \cap N_{\text{relevant}}}{N_{\text{relevant}}} = \frac{3213 \cap 25}{3213} = \frac{25}{3213} = 0,0077$$

Support Functionalities

Knowledge Analysis Support

In order to evaluate the knowledge analysis support, three queries focused on topics based on user interest (ISP network operators and IT managers, for example) were made.

The first query seeks to identify Brazilian ASes (Autonomous System) related to anomalous and malicious traffic and show the result by a domain structure graph (mind map). Figure 7.8 illustrates a domain structure of the term “ASN Brazil” in January 4th, 2010 (query example: **ASN Brazil –malicious ASN –date 01042010 to 01042010**).

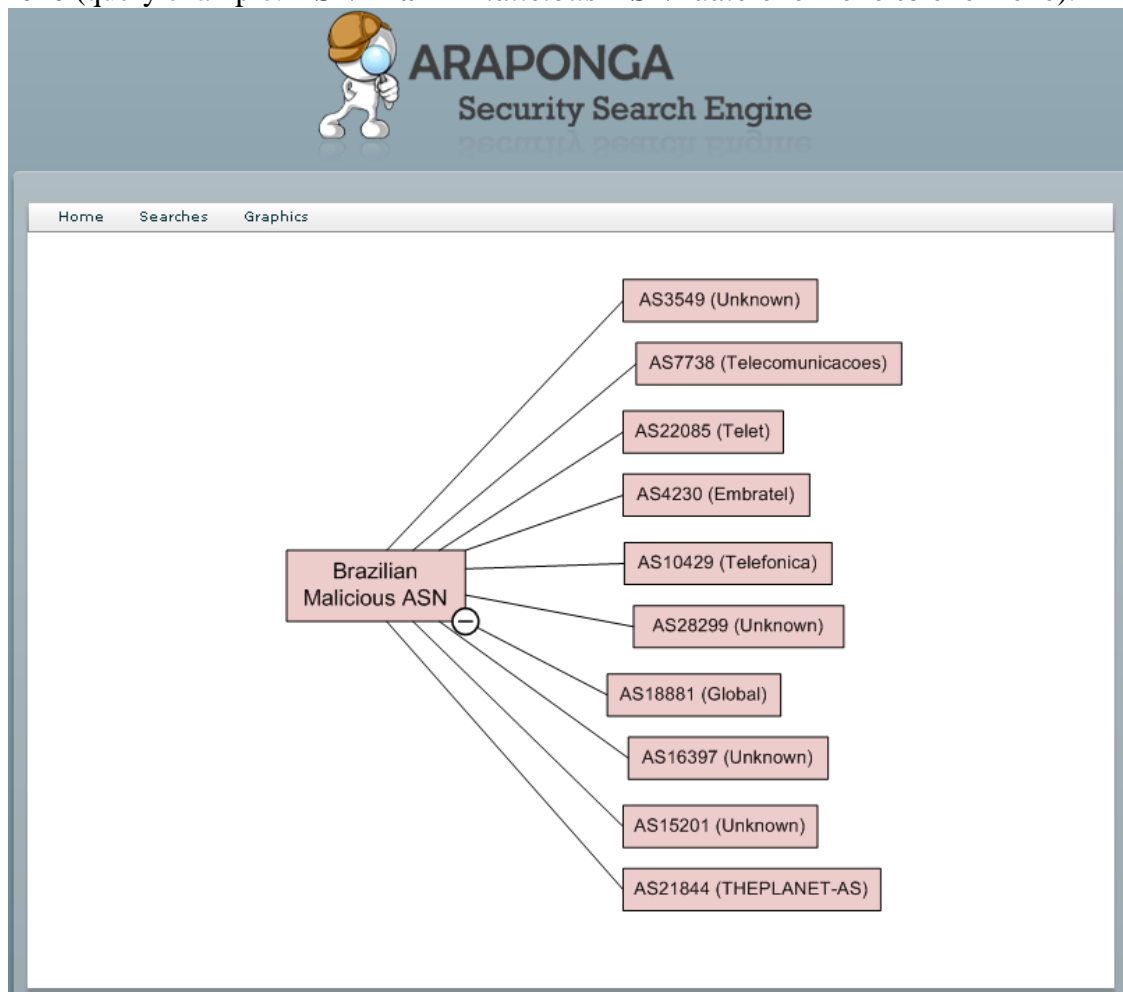


FIGURE 7.8: Brazilian ASes related to malicious activities

The second example query exemplifies an interest topic distribution of the Atlas [63] Web site, based on the identified page types during the indexing process. The goal is to provide an overview about the number of publications (referenced pages) involved attacks and malicious activity and consequently indicating the trend (increase or decrease) of determined type of content.

Figure 7.9 illustrates a frequency distribution graphic generated by ARAPONGA based on the query *-statistic -pageType atlas*.

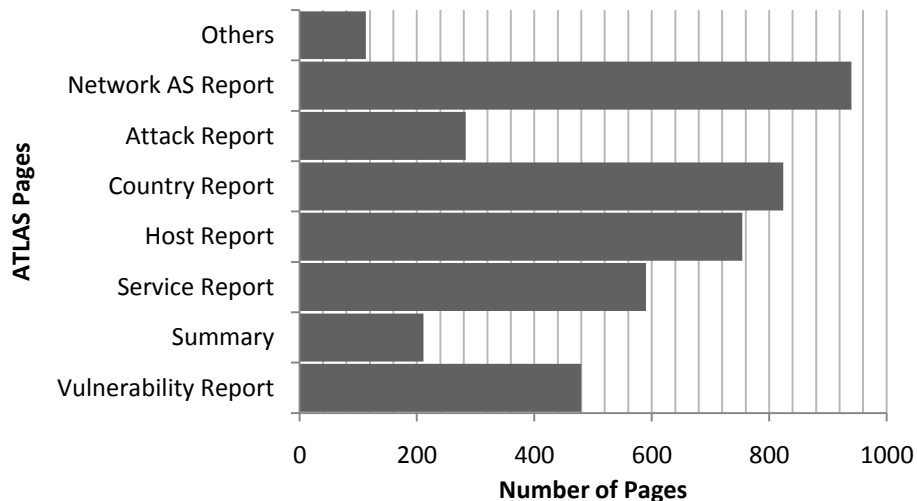


FIGURE 7.9: Frequency distribution in Atlas Web site

The third and last example query depicts the evolution of collected information. Its goal is to show a timeline of the apparitions of determined topic, permitting that the security team stays informed or pays more attention about the level of published pages of a specific topic. In this example, the query identifies the advertized vulnerabilities of the Microsoft Internet Explorer. The formulated query is as follows:

Microsoft Internet Explorer *-field system_affected,software -date 01112010 to 01152010.*

Figure 7.10 shows the graphic generated by ARAPONGA referent to the Microsoft Internet Explorer vulnerabilities found among January 11th to 17th, 2010.

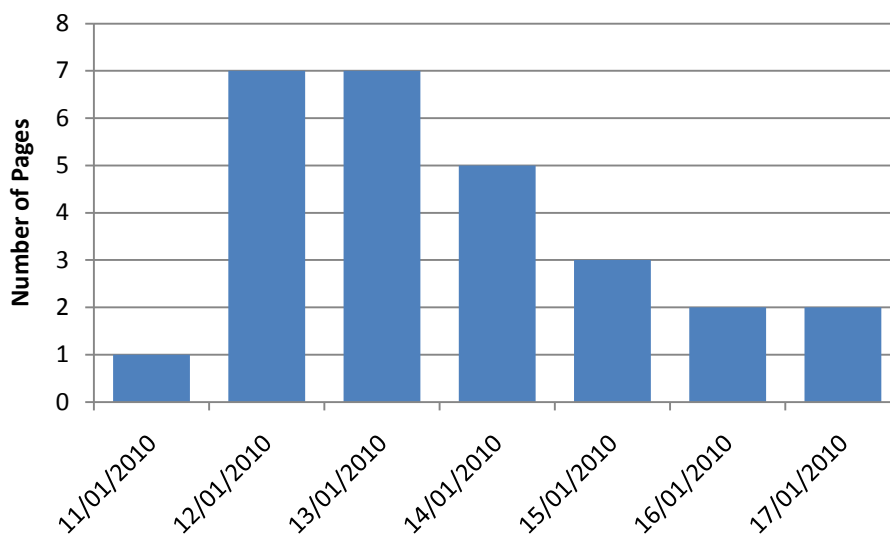


FIGURE 7.10: A timeline of Microsoft Internet Explorer vulnerabilities

7.5. Chapter Summary

This chapter presented a tool designed to obtain vulnerability information and Internet anomaly traffic statistics. It is a current proof of concept implementation that combines the use of data mining techniques and models (templates) to enhance the capacity of

indexing more accurately security information and consequently permitting differenced and more focused queries. The use of templates shows the advantages over blind search.

Towards this end, ARAPONGA makes the following contributions. The concepts of WIRSS and IRSS are applied to provide more supporting functionalities that transcend traditional search and browsing. It concentrates security information available from many sites into a unique base, containing only relevant data. Dozens of Web vulnerability reporting sites were evaluated in terms of completeness and access to content. The most adequate ones were selected for consultation by ARAPONGA.

ARAPONGA does not aim to provide a semantic search engine, but a small step is taken by this work towards the tailored retrieval, monitoring, management and user oriented visualization of Web information.

Expandir!!! (Paulo Cunha)

Part III

Results

Chapter 8

OADS Implementation and Performance Evaluation

This chapter describes a number of proofs of concept to progressive scenarios and the overall OADS-based implementation. More specifically, concrete performance examples are analyzed in terms of both their accuracy and response times. Firstly, the implementation of the architecture for anomaly detection, called OADS, is presented, including two new modules so far not formally described, namely: the Decision Service and the ADS-Fusion. Next, the adopted heuristic responsible for the orchestration and analysis of alerts is described. Then, some specially crafted testbed based scenarios are illustrated to emphasize OADS's role in the detection of unwanted traffic. Lastly, the results from such attack scenarios are shown and discussed.

8.1. OADS Implementation

As already explained in Chapter 4, the core of this thesis is based on making use the known concept of orchestration to explore the collaboration and harmonization among different anomaly detectors. In other words, the essence of the OADS framework lies in the power gained from the clever combination and coordinated orchestration of different attack detection modules. As a proof of that, the OADS prototype combines modules from a famous and proven reliable IDS tool (Snort [163]) with two previously introduced detection strategies (ChkModel [158] and Profiling [151]).

Figure 8.1 illustrates the architectural organization of OADS and shows its components.

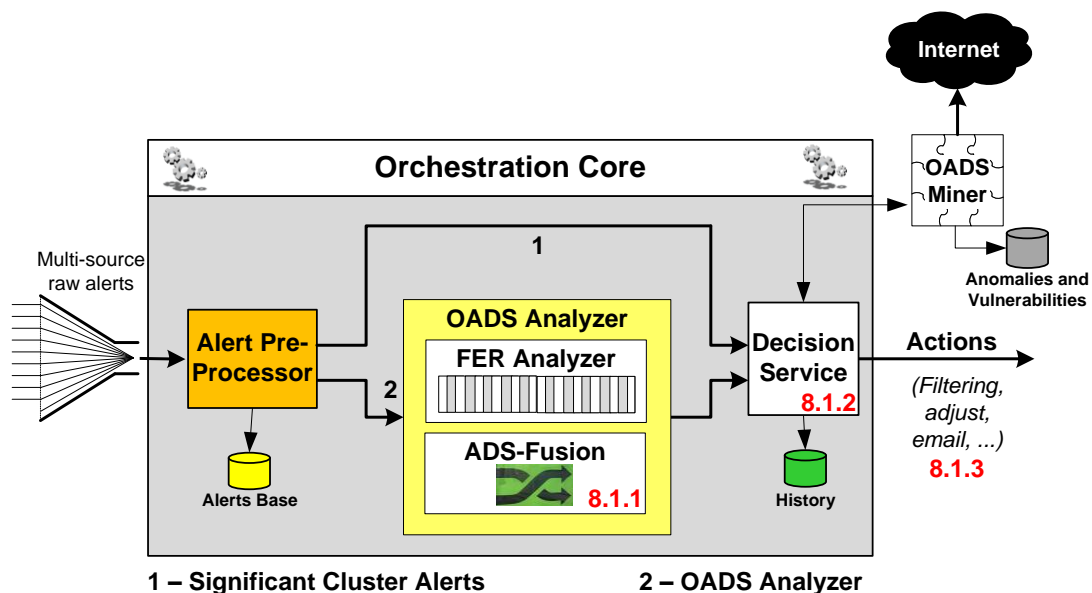


FIGURE 8.1: OADS implementation architecture

The following sub-sections explain in details each of the component of the OADS architecture.

8.1.1. ADS-Fusion

Another module used to support the OADS Analyzer component is referred to as ADS-Fusion. The central idea is to employ data fusion techniques to deal with uncertainty or imprecision of anomaly detection results and consequently increase the degree of confidence about intrusive or malicious activities, allowing that more correct and accurate actions can be taken.

ADS-Fusion [224][225] is based on Dempster-Shafer's Theory of Evidence (DST) [226][227][228]. DST is one of most known mathematical models to represent uncertainty in knowledge-based systems. It focuses on solving problems and modeling uncertainty when using purely probabilistic methods. Unlike other Bayesian probabilistic theory, DST does not need prior knowledge of the probabilistic distributions of the studied elements. This allows attributing belief values - Basic Probability Assignment or simply *bpa* in DST – for a subset of possibilities and not only for simple events.

Another important feature is that the belief not assigned to any event in particular but attributed to the environment and not to the rest of the evidence. Moreover, it is possible to combine belief functions for generating new functions of belief in an independent manner of the order of appearance of new evidence requiring only that the original assumptions are mutually exclusive and exhaustive.

ADS-Fusion architecture

ADS-Fusion was designed to study anomaly detection techniques and develop a system capable of increasing the efficiency of detection through the data fusion. Practically speaking, it is a software module that receives the outputs generated by anomaly detectors as input, makes data fusion of these inputs, and produces an inference with a greater degree of certainty than the uncertainty generated by anomaly detectors individually. ADS-Fusion was originally implemented in C++, but currently it is deployed in Java.

ADS-Fusion is composed by three elements: collector, sensors, and data fusion engine. **Collector** is responsible for capturing the network traffic and generating trace files in standard format. **Sensors** components are responsible for analyzing data generated by the collector and detecting possible anomalies. Another key role of these sensors, in the context of this additional mechanism, is assigning a belief for each generated inference. The **Data Fusion Engine** is responsible for decision making. It uses DST combination rules to associate and correlate the different analysis and results of distinct sensors to generate more accurate inferences with a greater degree of accuracy.

In the current OADS implementation, there is no need for the two ADS-Fusion components collector and sensor. Instead, the ADS-Fusion module receives such information from OADS Alert Pre-Processor. It is important to explain its important role as a hypotheses generator about the possible real network state. Hence, in order to work with Profiling, ChkModel and Snort detectors, it is necessary to adjust the alerts generated by them, aiming to assign a well calibrated *bpa* value for each alert type. Details of how this is made are explained below.

ChkModel

The *bpa* generation in ChkModel [158] is based on the distance among obtained values from the *Threshold Adaptive* function. When the return of such function is equal to 1,

ChkModel considers the Network State as “Under Attack” and then calculates the *bpa*. The greater the distance between the values of obtained threshold and the established threshold, the greater is the belief in the existence of an attack.

Considering the example with an output that contains the detection of an anomalous connection between 58.33.126.229 and 192.168.0.163 IP addresses, where the threshold of packets exchange was calculated in 6. Supposing that for this network the threshold for “Normal” state is equal to 5, any connection that is above this limit will be considered anomalous. By fixing the belief of the normal state at 0.5, it is possible to determine the belief of the attack as being the sum between belief and the normal rate of increase ($6 / 5 = 1.2$, which represents a percentage increase of 20%), obtaining thus a *bpa* = 0.6

Profiling

The *bpa* generation in Profiling is offered evaluating the BCs, the frequency of repetition between them and the quantity of flows associated with this classification. For example, considering *dstIP* as cluster key, at the first interaction (default time slot) the IP destination 10.108.40.X (150 flows) is classified as BC = 24 (DDoS attack for key group). On the second interaction, the BC for this IP remained the same and the number of flows increased to 450, then it is possible to increase the belief on this inference. Fixing the belief of the normal state at 0.5, it is possible to determine the belief of the attack as the sum between belief and the normal rate of increase ($450 / 150 = 3$, which represents a percentage increase of 200%), obtaining thus a *bpa* = 1.0, the maximum possible value.

Snort

The *bpa* generation in Snort is based on severity field present in its alerts. For this, initial beliefs for each possible value of this parameter are established as follows: Low corresponds to 0.5, Medium corresponds to 0.65 and High corresponds to 0.8. In addition, the frequency of repetition is also considered to increment these values.

For example, in a sequence of alerts, the first five represent the same attack and have same features, including severity level equal to low. This way, the first alert to be evaluated will have an attribute *bpa* equal to 0.5. The second will increase the *bpa* in 0.01. Now the *bpa* is 0.51. The next three alerts also will be increase the *bpa*. After the five alerts have been evaluated, their *bpa* is 0.55. For alerts with severity level equal to medium e high, the degree of increase is 0.05 and 0.1, respectively.

Data Fusion Engine

The Data Fusion Engine component was implemented using JDS [229], an API written in java that supports the basic function of DST such as belief function (*bel*) and plausibility (*pl*).

Overall, this engine’s operation can be summarized in three stages:

- **Synchronization:** this process consists in establishing links (connections) among the related events to both sensors. It is an essential process because it is necessary to ensure that each of the events to be combined refers to the same time interval. For example, if at any one time *t*, the ChkModel detects a DDoS attack, it is important that this event is then combined with the Profiling analysis of the same time *t* so that the ADS-Fusion can generate a more accurate inference. This stage requires some small changes in the generation of results from the sensors.

- **Combination:** this stage is used as “clues” for the generation of inferences from the current network state. It combines (or transforms) the inputs into JDS elements, which basically contain the state identified (LOW, MEDIUM and HIGH) and the belief level in this state (*bpa*).
- **Inference Generation:** this stage determines the severity level of the anomaly or anomalies affecting the network.

8.1.2. Decision Service

The Decision Service component is responsible for the decision process related to analyzed network traffic. According to received information (or even collected), it establishes what to do next.

It receives two types of inputs. The first one is a set of reduced alerts, sent by Alert Pre-Processor, whereas the second one is rules, sent by the OADS analyzer (FER Analyzer and ADS-Fusion modules). Although ultimately both inputs have the same structure composed by a source IP address, source port, destination IP address, destination port, class of attack and its severity, the treatment given to each is different. This is because, unlike the rules, the reduced alerts are typically received first and contain more reliable information, due to the process of significant cluster extraction. This is in contrast with the rules that are the result of extensive analysis performed on apparently less relevant data.

For such reason, the Decision Service adopts two different strategies: one to deal with reduced alerts and second one for rule processing. The former employs the same idea used in Chapter 6 to demonstrate the applicability of episode rules (see section 6.3.2). It translates alerts to basic firewall rules for enforcement at other devices. Recall that enforcement actions are not the focus of this thesis.

As far as is concerned the use of rules as input, a simple finite state machine is used. It correlates the information processed by different analyzers in possible states that the decision service could take.

Table 8.1 shows the machine states that can be assigned according to the used detectors.

TABLE 8.1: Representation of information sent by detectors to Decision Service

Profiling	ChkModel	Snort	State
<i>Good</i>	<i>Good</i>	<i>Good</i>	<i>State-A</i>
<i>Good</i>	<i>Good</i>	<i>Low</i>	<i>State-B1</i>
<i>Good</i>	<i>Good</i>	<i>Medium</i>	<i>State-C1</i>
<i>Good</i>	<i>Good</i>	<i>High</i>	<i>State-D</i>
<i>Good</i>	<i>Suspicious</i>	<i>Good</i>	<i>State-B1</i>
<i>Good</i>	<i>Suspicious</i>	<i>Low</i>	<i>State-B2</i>
<i>Good</i>	<i>Suspicious</i>	<i>Medium</i>	<i>State-C1</i>
<i>Good</i>	<i>Suspicious</i>	<i>High</i>	<i>State-D</i>
<i>Good</i>	<i>Bad</i>	<i>Good</i>	<i>State-C1</i>
<i>Good</i>	<i>Bad</i>	<i>Low</i>	<i>State-C1</i>
<i>Good</i>	<i>Bad</i>	<i>Medium</i>	<i>State-C2</i>
<i>Good</i>	<i>Bad</i>	<i>High</i>	<i>State-D</i>
<i>Bad</i>	<i>Good</i>	<i>Good</i>	<i>State-C1</i>
<i>Bad</i>	<i>Good</i>	<i>Low</i>	<i>State-C1</i>

<i>Bad</i>	<i>Good</i>	<i>Medium</i>	<i>State-C2</i>
<i>Bad</i>	<i>Good</i>	<i>High</i>	<i>State-D</i>
<i>Bad</i>	<i>Suspicious</i>	<i>Good</i>	<i>State-C2</i>
<i>Bad</i>	<i>Suspicious</i>	<i>Low</i>	<i>State-C2</i>
<i>Bad</i>	<i>Suspicious</i>	<i>Medium</i>	<i>State-D</i>
<i>Bad</i>	<i>Suspicious</i>	<i>High</i>	<i>State-D</i>
<i>Bad</i>	<i>Bad</i>	<i>Good</i>	<i>State-D</i>
<i>Bad</i>	<i>Bad</i>	<i>Low</i>	<i>State-D</i>
<i>Bad</i>	<i>Bad</i>	<i>Medium</i>	<i>State-D</i>
<i>Bad</i>	<i>Bad</i>	<i>High</i>	<i>State-D</i>

Note that the information relative to the classification (good, bad, transient, low, medium and high) is transported by IDMEF messages into a severity field. Figure 8.2 depicts the mapping between these information and the enforcement actions.

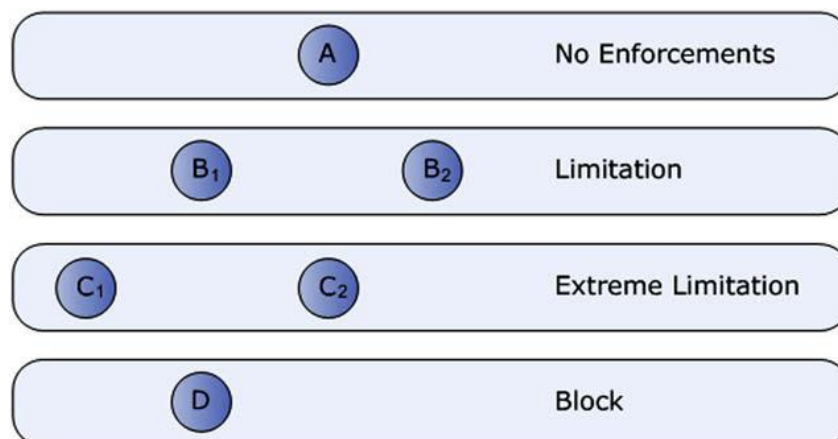


FIGURE 8.2: Mapping classifications and enforcement actions

As mentioned in section 6.3.2, possible values to the enforcement actions are: Limitation, during 60 seconds; Extreme Limitation, during 300 seconds; and Blocking of given traffic.

8.1.3. Enforcement Actions

In order to prove OADS functionalities, some enforcement component responsible for receiving decisions from the Decision Service and translating these into real network actions are defined. As previously described in Chapter 6 (see section 6.3.2), this is actually represented by an enforcement agent that translates the decisions to firewall rules, specifically for IPTables.

8.2. Heuristic for Orchestration

This work employs the concept of orchestration to automatically manage the inputs of different anomaly detectors, harmonize them and consequently be able to make a well informed, correct and efficient decision with regard to the existence of traffic anomalies. Throughout the text, there is mention of the likely solutions for stopping and mitigating unwanted traffic though this is not the object of this thesis. The present work concentrates its efforts on the detection phase of unwanted traffic and the mechanisms for that. Before going into the details of the OADS implementation, it is important to shed some light on the way the concept of orchestration is used in the context of this work.

Formally, orchestration refers to an executable business process that may interact with both internal and external complex computer systems, middleware, and services. Currently, orchestration is mainly related to connecting Web services in a collaborative fashion. Orchestration establishes the sequence of steps within a process, including conditions and exceptions, and creates a central controller to implement the sequence. (referencia)

Though OADS may adopt Web Services as its underlying coordination engine to coordinate its actions within a Web environment, it also operates on a standalone manner. It is this second implementation mode that is emphasized throughout the rest of this document. Typically, the use of emerging Internet standard effort as the Business Process Execution Language for Web Services (BPEL4WS) [230] and the Service Oriented Architecture (SOA) [231] has the clear advantage of opening access to an unlimited number of services and security applications that adopts such technologies. At the time of this work, there is still limited adoption of Web Services and as such, one does not see the need for the added complexity toward OADS design.

Despite the above, the orchestration concept itself remains of interest to OADS and as such it is present at its heart, managing all the interactions among its applications (preprocessors and analyzers) in a controlled way. A simple, parameterized and effective heuristic (represented as an algorithm) mimicking orchestration is used instead. Such heuristic is based on common rules and acquired knowledge that governs the way the received information (alerts and rules) should be treated by the OADS core. Despite its simplicity, this intelligent module obtains results as shown in latter sessions of this chapter.

Next, details of the orchestration heuristic are described in Algorithm 8.1.

Algorithm 8.1 Simplified algorithm for orchestration

Step 0: Initialization

Read alerts every x time

While TRUE

Step 1: *Alert Pre-Processor* receive all multi-source alerts

All received alert are prepared for possible extraction of significant cluster process.

Step 2: Significant cluster extraction

Execute significant cluster extraction process.

All alerts classified as significant are send to the *Decision Service* (**Step 6**).

Step 3: *If* there are alerts to evaluate, go to **step 3a**, otherwise go to **Step 6**

Step 3a:

If number of remain alerts > *fer_threshold*, go to **step 4**;

Otherwise go to **step 5**

Step 4: FER Analyzer examines received alerts

For all alerts

Calculate Frequent Episodes.

Generate Rule Episodes.

End

Send all rules to *Decision Service* (**Step 6**).

Step 5: ADS-Fusion examines received alerts

If alerts have two or more sources, go to **step 5a**, otherwise go to **Step 6**

Step 5a:

Execute Dempster-Shafer analysis in alerts.

Send all inferences (rules) to *Decision Service* (**Step 6**)

Step 6: Decision Service receives alerts or analysis results

Evaluate the received information

If necessary use OADS Miner to discover extra information
 Make decisions

End

The above six steps are explained. All received multi-source alerts are handled by the Alert Pre-Processor component which extracts all the required alert attributes for analysis (step 1). Next, the extraction of significant clusters is started (step 2). As previously described in Chapter 6, the idea is to extract significant information from clusters of interest (*srcIP*, *dstIP* and *class*). The output of this process (in the form of classified alerts) is directly sent to the Decision Service component. It decides what to do next (step 6).

Although the studies in Chapter 6 demonstrate that the automatic identification of relevant information is successful, it is also possible that some alerts, or even all of these in some cases, are not considered relevant enough to take any decision. However, instead of simply ignoring these, the adopted orchestration heuristic follows a series of selective procedures, in an attempt to make use of these and likely improve the current analysis. Such steps are:

1. First, to verify if there are any alerts classified as not relevant subsequent to the significant cluster extraction process. If none are encountered, meaning that all alerts are significant, the algorithm proceeds directly to Step 6, where the Decision Service component must process all such information in order to reach one or more decisions. Otherwise, the current amount of alerts is compared with a pre-defined threshold (*fer_threshold*), used to evaluate the viability of performing a FER analysis over these alerts (Step 3). The studies about FER analysis from Chapter 6 show that the smaller is the number of events, the less is the probability of detecting any frequent episodes. Consequently, some important considerations are needed with regard to configuring FER analysis as shown in Table 8.2.

TABLE 8.2: FER parameters for orchestration algorithm

Number of Alerts (<i>fer_threshold</i>)	Window Size	Frequency Threshold	Confidence Threshold
>50	10	0.01	0.80
>500	10	0.02	0.80
>5000	20	0.05	0.80

In the case where the number of alerts is greater than the threshold *fer_threshold*, the FER Analyzer component receives these alerts (Step 4). Otherwise, the remaining alerts are sent to the ADS-Fusion component (Step 5).

2. Under FER Analyzer (in Step 4), the alerts are processed to discover the existence of frequent episodes. These are calculated and episode rules are possibly generated next. This process has been extensively explained in Chapter 7. Lastly, the episode rules are sent to Decision Service component according to Step 6.
3. During the ADS-Fusion analysis (shown as Step 5), the alerts are evaluated using the known Dempster-Shafer evidence theory. This is used to reduce the uncertainty of these alerts and increase their degree of confidence. However, before undertaking this analysis, the received alerts are verified to determine if they are descendant of two or more distinct detectors, a

requirement of DST analysis. If this is not case, the alerts are forwarded to the Decision Service. Otherwise, they are processed and the obtained inferences (or rules) are then sent to the Decision Service.

The last step in this important heuristic is performed by the Decision Service. To put it more simply, it is fed with a diversified set of inputs, including alerts, episodes rules and inferences which it must process before reaching any decision. The present proposal also includes an additional auxiliary mechanism. This is in the form of an OADS miner module, ARAPONGA. It is used to obtain more information about some alerts over the Internet. Finally, a decision is generated.

It is important to emphasize that, as described in Chapter 4, OADS's approach allows a number of different types of reactions. Under the present implementation, two main decisions are supported. The first one consists of limiting or mitigating something considered malicious and bad by blocking its traffic. This decision is the most common one and usually is taken by the Decision Service component. The second one consists of simply not be taking any action. This lack of decision may be the case when there is insufficient certainty (evidence) to act upon.

8.3. Testbed environment

OADS testing was purposely confined to an isolated testbed consisting of real machines within GPRT laboratory. The idea was to create a controllable environment that resembles as much as possible a realistic network topology that can be subjected for example to DoS and DDoS attacks. As depicted by Figure 8.3, this testbed contains around 60 PCs, 3 Cisco switches with 24 and 48 10/100/1000 Mbps interfaces. The PCs are used as edge nodes running different user applications, to simulate routers, and application level traffic generators. They are running different operational systems, especially those from the Windows family (XP, Vista and 7) and Ubuntu Linux. The PCs attackers also run similar operating systems.

The OADS server is an Intel Core 2 Quad CPU, with 4 processor Q6600 (2.40 GHz), 4Gb of RAM, 500 GB of HDD and one network interface 10/100/1000. Although all OADS components (Alert preprocessor, FER Analysis, DST Analyzer and Decision Service) were designed and implemented to work in a distributed setup, they are collocated in the testbed server, except for the OADS Miner which requires important resources for crawling the Internet for updating its security information base.

Regarding detectors, a range of these was used including: various Snort [163] version 2.8.6, ChkModel [158] and Profiling [151]. As depicted by Figure 8.3, these detectors are spread across specific interest points of the network. All servers run Linux distributions, including Ubuntu and Debian. Firewall/Router server is running the FreeBSD operational system. Table 8.3 describes the localization and type of employed detectors in the testbed.

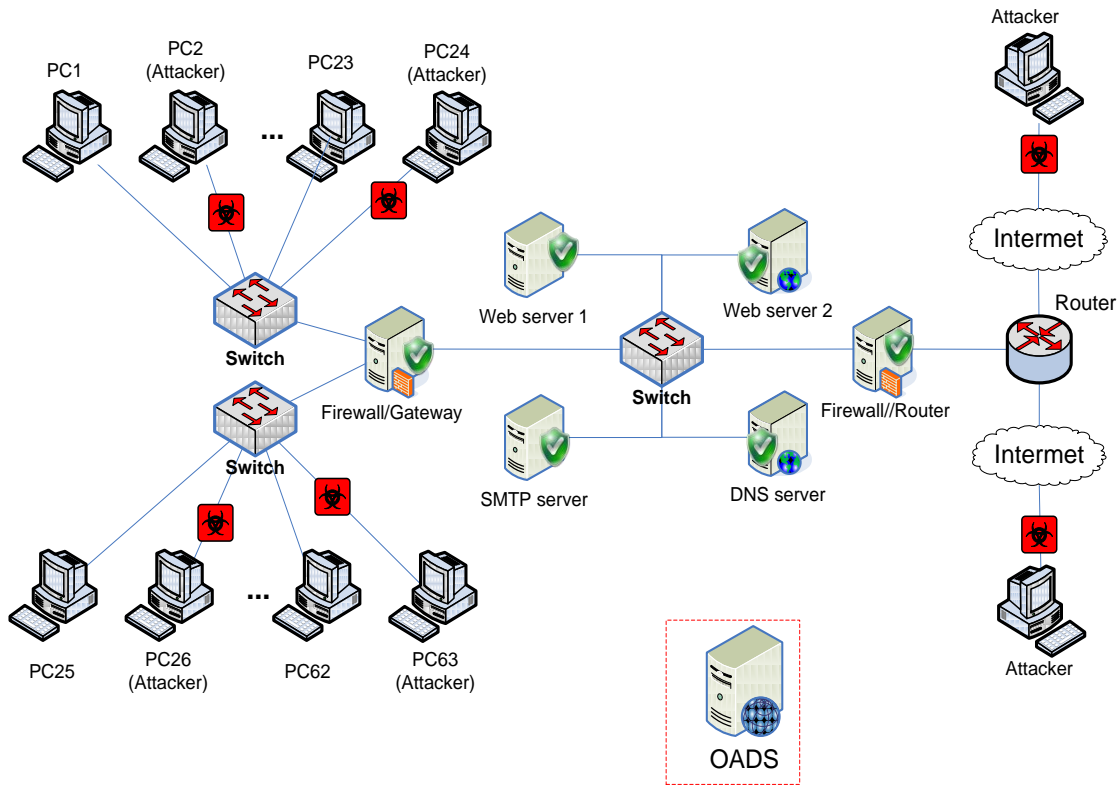


FIGURE 8.3: Full OADS Testbed Topology

TABLE 8.3: Distribution of detectors in OADS testbed

Server	Detector
<i>Firewall/Gateway</i>	Snort (default configuration)+ Profiling + ChkModel
<i>DNS server</i>	Snort (DNS configuration) + Profiling (1 minute configuration)
<i>SMTP server</i>	Snort (default configuration) + Profiling (1 minute configuration)
<i>Web server 1</i>	Snort (default configuration) + ChkModel
<i>Web server 2</i>	Snort (emergent configuration) + ChkModel
<i>Firewall/Router</i>	Snort (default configuration)+ Profiling + ChkModel

It is important to emphasize some aspects of these detectors. Most Snort detectors were set up to execute only with default configuration, provided by Snort manufactures. Two different Snorts were set up to detect specific attacks. The first one, located on a DNS server, had a configuration adequate for detecting DNS attacks and anomalies provided by Emergent Threats [232]. The second one, located on a Web server 2, had a configuration adequate for the detection of Web attacks as well as anomalies also provided by Emergent Threats.

Regarding the ChkModel [158], it was designed and used to evaluate only TCP packets and cannot be used to analyze UDP attacks. Finally, two different Profiling configurations were used. Though both followed the original specifications [151], the first one was set up to perform evaluations over a time interval of five minutes (according to the proposal) whereas the second one was set up to operate over time intervals of one minute. Obviously, this difference will reflect on the number of generated alerts during the analysis.

8.3.1. Malicious traffic generation

In order to test different attacks and anomalies, two types of solutions were deployed: a tool for packet injection and a range of python scripts. The former, called Packet Analysis and Injection Tool or simply Packit [233], is a network tool designed to customize, inject, monitor, and manipulate IP traffic. It allows the spoofing of nearly all TCP, UDP, ICMP, IP, ARP, RARP, and Ethernet header options. Packit is useful for testing firewalls, intrusion detection/prevention systems, port scanning, simulating network traffic, and general TCP/IP auditing. Packit was used to create customizable DoS and DDoS attack scripts.

The latter is a set of python scripts using Scapy [234], a powerful interactive packet manipulation python library. Scapy is able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies, and much more. It can easily handle most classical tasks like scanning, tracerouting, probing, unit testing, attacks or network discovery. In addition to these solutions, Internet script to perform Slowloris HTTP DoS [235] attacks was also used.

8.4. Orchestrating Analysis

The essence of the OADS approach lies in the power gained from the clever combination and coordinated orchestration of different attack detection modules. In order to cover the variety of attacks and to fairly evaluate the robustness of this work, different attack scenarios are planned, aiming to illustrate the efficacy and more importantly attack detection of the AODS solution.

It is important to emphasize that the analysis and evaluation process is based on the steps of the orchestration algorithm. Next some of the experiments are described.

8.4.1. Scan UPnP

The first experiment for analysis can and must be considered an unplanned event. It is a residual traffic collected by Snort detector, located on Firewall/Gateway computer, during the third initial minutes of monitoring while preparing for a DNS cache poisoning attack (second experiment).

Overall, Snort (Firewall/Gateway) sends 61 alerts reporting “*SCAN UPnP*” service discovery from a GPRT laboratory computer (150.161.192.X) to the Internet (239.255.255.250), during thirty minutes. Figure 8.4 illustrates the time line of these alerts.

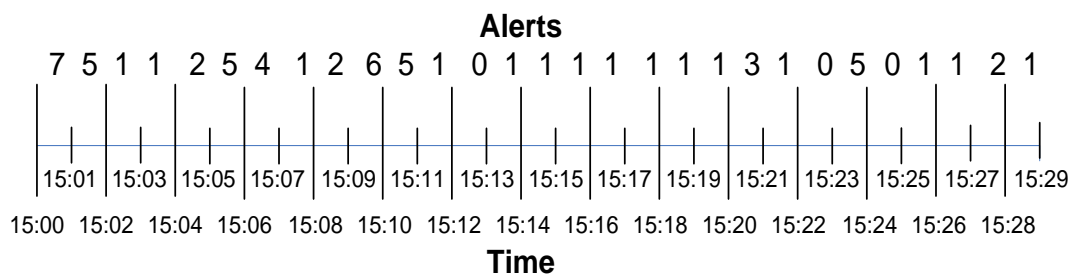


FIGURE 8.4: UPnP alerts time line

Analysis

As established in Algorithm 8.1, for each time interval the orchestration algorithm must evaluate the received alerts and make decisions about what course of action to take. At 15:01, 7 alerts are received for the Alert Pre-Processor component for handling alerts (step 1) and are next forwarded for significant clusters extraction (step 2). As previously described in Chapter 5, the process begins with the calculation of RU considering all elements inside the set (in this case, 7 alerts). The results are $RU(A) = 0.502345126754$ for *class*, *srcIP* and *dstIP* cluster. Consequently, these alerts were considered significant, since the RU value is less than β (0.9).

Thus, according to Algorithm 8.1, the next step consists of sending the alerts to the Decision Service (Step 6). This service receives all alerts and employs its state machine in order to evaluate them. As explained in Table 8.2, the states correspond to the combination of the alerts source detectors and the level of severity of each one. In this case, all alerts have the same origin and same severity (high). The result of the state machine analysis is the transition to state **D**, since it was assumed that it is case where Profiling is *good*, ChkModel is *good* and Snort is *high*. Ultimately, this decision can be translated to the following action: block any packet sent by IP address 150.161.192.X, source port 56134, destined to IP address 239.255.255.250 with destination port 1900. Figure 8.5 shows a simple example for IPTables.

```
IPTABLES -A OUTPUT -s 150.161.192.52 --sport 56134 -d 239.255.255.250 --
dport 1900 -j DROP
IPTABLES -A OUTPUT -d 239.255.255.250 --dport 1900 -j DROP
```

FIGURE 8.5: Possible IPTables rules for UPnP alerts

Since two additional identical alerts are received in the next times (15:02 and 15:03), the same evaluation process is followed until the step 6. However, as described in Algorithm 8.1, Decision Service makes use of OADS Miner (ARAPONGA) to obtain more information about this type of alert.

For this, it makes the following query: SCAN UPnP service discover attempt - *summary_vulnerability*. This query goes through all indexed contents looking for the searched term(s). It then returns an XML file containing the following information according to Figure 8.6. The page states that this vulnerability allows an intruder to run code over an invaded machine and suggests the application of a patch for this type of known attack and disable UPnP.


```

<Summary_Vulnerability>
  <Vulnerability id="1">
    <Title name="SCAN UPnP Service Discovery Attempt">
    <Description>
      Universal Plug and Play (UPnP) is a system to allow network devices to operate together. A
      vulnerability in the Microsoft Windows XP and Windows ME implementation of UPnP may permit an intruder
      to execute arbitrary code with SYSTEM privileges. Additionally, Windows 98 and Windows 98SE may be
      affected if you have installed the Windows XP Internet Connection Sharing client.
    </Description>
    <Timeline>
      <Disclosure_Date>2001-12-20</Disclosure_Date>
      <Published_Date>2001-12-20</Published_Date>
      <Last_Update>2001-12-20</Last_Update>
    </Timeline>
    <Classification>
      <Attack_From>Remote</Attack_From>
      <Impact risk="high">An intruder can run arbitrary code in the local SYSTEM security context</Impact>
      <CVSS>7.5</CVSS>
      <CVE_ID>CVE-2001-0876</CVE_ID>
    </Classification>
    <Affected>
      <product>Windows XP</product>
      <product>Windows ME</product>
      <product>Windows 98</product>
    </Affected>
    <Solution>
      To disable UPNP.
      Apply a patch as described in MS01-059.
    </Solution>
    <References>
      <name>Microsoft Security Bulletin: MS01-059</name>
      <url>http://www.microsoft.com/technet/security/bulletin/ms01-059.asp</url>
      <name>CERT/CC vulnerability note: VU#951555</name>
      <url>http://www.kb.cert.org/vuls/id/951555</url>
      <name>Computer Incident Advisory Center Bulletin: M-030</name>
      <url>http://www.ciac.org/ciac/bulletins/m-030.shtml</url>
    </References>
  </Vulnerability>
</Summary_Vulnerability>

```

FIGURE 8.6: Summary_vulnerability for SCAN UPnP alert

8.4.2. DNS cache poisoning

DNS cache poisoning is an attack that consists of changing or adding records to the resolver caches, either on the client or the server. The objective is so that a DNS query for a domain returns an IP address for an attacker's domain instead of the intended domain. According to Hyatt [236], DNS cache poisoning results in pharming, which allows the attackers to perform identity theft, distribution of malware, dissemination of false information, and man-in-the-middle attacks.

The Experiment

The current DNS cache poisoning experiment aims to add a new domain named *feitosa.tnt* into the authoritative DNS server of GPRT laboratory. In order to achieve this goal, two computers are used (all of them running Linux distribution). They both execute a python script (called *DNScache poisoning.py*) that exploits the vulnerability

discovered by Dan Kaminsky [55]. This script sends fake recursive queries aiming to insert a dummy record in the vulnerable DNS server by guessing the transaction ID. It also inserts an Authority record for a valid record of the targeted domain. The script uses a random source IP address, a source port number equal to 32883 (the vulnerable DNS port for recursive queries) and the transaction ID starting with 1024 and increasing +1 for each interaction. Figure 8.7 illustrates this scenario.

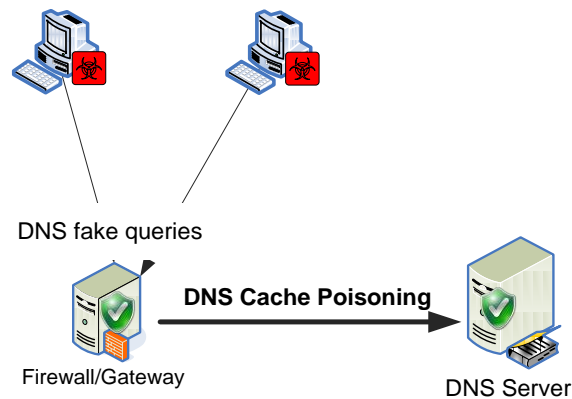


FIGURE 8.7: DNS cache poisoning scenario

In order to clarify this attack experiment targeted to GPRT DNS server, Figure 8.8 clearly shows the increase in the packet number seen before and after the attack is started.

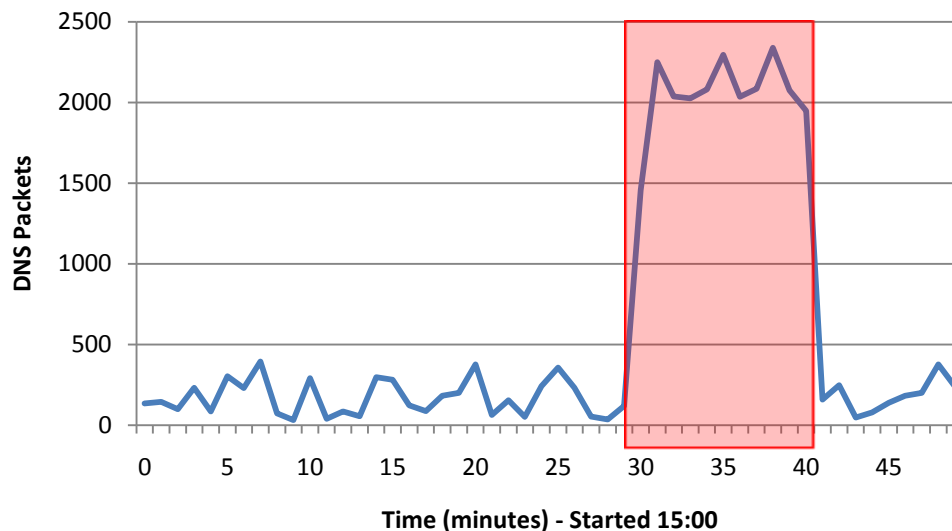


FIGURE 8.8: DNS cache poisoning attack without defense

The line plotted in the graphic depicts the amount of UDP DNS packets during a time interval between 15:00 and 15:50 of June, 02 2010, without the presence of any type of defense. Before the attack taking place, there was a mean of 167 packets per second whereas it suddenly increased to 586 packets per second once the attack was launched.

The first thirty minutes corresponded to normal user traffic while all detectors were running and without injecting any attack traffic. At the thirtieth minute, attacks are injected into the local area network. Hence, from now on, alert classification, evaluation

and decision start taking place. Ten minutes into the experiment, the attack traffic is halted.

Analysis

The DNS cache poisoning attack took place at 15:30. As shown in Figure 8.8, the average of received packets per second suddenly increased from 167 to 586 packets per second once the attack was launched.

During the first minute of the attack (15:31), Alert Pre-Processor received two alert files, in the IDMEF format. These represent alerts from the Snort (Firewall/Gateway) and Snort (DNS server), containing 6864 and 30 alerts respectively. The Pre-Processor has therefore to perform significant cluster extraction. On the other hand, ChkModel did not generate any alerts because it only supports TCP inspection and not that of UDP packets, and Profiling only generated alerts after two time slots (in this case, two minutes).

The Snort detector (Firewall/Gateway) generated a huge number of alerts to type “*DNS response for RFC1918*”, according to the original DNS rule. The other Snort detector (DNS server) generated alerts classified as “*ET CURRENT_EVENTS DNS Query Responses with 3 RR’s set (50+ in 2 seconds) – possible A RR Cache Poisoning Attempt*”, according to Emergent threat’s [230] DNS rule file. The difference of the number of alerts between both Snort detectors is due to the fact that the first one is located into the local LAN segment of the GPRT network (that uses NAT IP addresses) and therefore receives directly the attack. On the other hand, the second Snort detector running at the DNS server is located on a DMZ segment and only receives the attack once it passes through the two hosts (Firewall/Gateway of LAN segment and Gateway of DMZ segment). Therefore, this reduces the number of packets received by the second Snort detector. When the second Snort generates one alert every 2 seconds, the first one generates between 5 and 8 alerts per second. Such discrepancy reflects the different rules employed in these detectors.

As far as is concerned to the orchestration analysis (Algorithm 8.1), all Snort alerts were considered significant, according to the three keys: *srcIP*, *dstIP* and *class* (all of them had an $RU(A) = 0.303334312$), and were consequently forwarded to the Decision Service. After receiving these alerts, the Decision Service performs a simple validation of these and eliminates those are duplicated. Consequently, only two alerts were analyzed and the following decisions were made:

- To block any packet sent by IP address 150.161.192.253, source port 53, destined to DNS server with destination port 32883 and;
- To block any packet sent by IP address 192.168.0.7 with source port 53, destined to 150.161.192.2:32883.

Figure 8.9 shows a simple example for IPTables configuration to achieve these two actions at the firewall.

```
IPTABLES -A INPUT -s 150.161.192.253 --sport 53 -d 150.161.192.2 --dport 32883
-j DROP
IPTABLES -A INPUT -s 192.168.0.7 --sport 53 -d 150.161.192.2 --dport 32883 -j
DROP
```

FIGURE 8.9 Possible IPTables rules for DNS cache poisoning alerts

At the third minute of the attack (15:02), Alert Pre-Processor received 1 alert file from the Profiling (DNS server) and 44 alerts from Snort (Firewall/Gateway). The Profiling alert represents the increase of the number of flows from IP address 150.161.192.253 (gateway) to IP address 150.161.192.2 (DNS server), as previously detected by the Snort (DNS server). It is important to emphasize that the Profiling detector (DNS server) needs at least two minutes to start generating alerts and its output is naturally summarized, what explain having only a single alert.

The alerts of Snort (Firewall/Gateway) are of the same classification as those from the previous (*DNS response for RFC1918*). However, the amount of generated alerts is smaller. Such fact is directly related with the enforcement of the decisions illustrated earlier on Figure 8.9 by the firewall. These block all packets from 150.161.192.253:53 to 150.161.192.2:32883 and all packets from 192.168.0.7:53 to 150.161.192.2:32883 from passing.

Considering the orchestration analysis, all Snort alerts were taken to be significant and sent to the Decision Service when the Profiling alert was evaluated to determine if it could be applied in FER analysis. However, as it does not fulfill the minimal threshold condition (number of alerts $>$ *fer_threshold*, imposed by step 3a of the orchestration algorithm), no forwarding was made. The Decision Service suggested blocking any packet sent by IP address 150.161.192.253 with source port 53 destined to DNS server on 32883 port.

Figure 8.10 gives a closer view of all attack. The average number of packets before this attack was 167 per minute and increased to 1456 packets per minute in the first minute of the attack. With the evaluation and decisions taken by the OADS heuristic, the attack effects were felt only the first 120 seconds. This time represents the average time that the architecture requires to detect and take an action to mitigating it. From then on, only the “normal” packets are seen in the network.

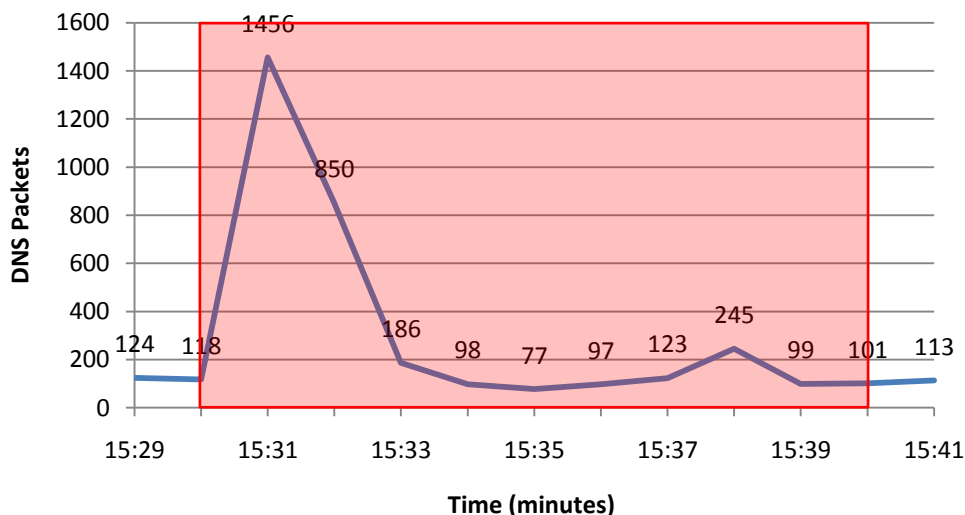


FIGURE 8.10: DNS cache poisoning attack with OADS decision and actions

Note that the decisions taken by OADS (via Decision Service) are applied at the two firewalls of the testbed topology. Consequently, Profiling and Snort, located on DNS server, stop to generate alerts during the rest of attack, since the internal traffic with source port 32883 was blocked from getting into the DMZ zone. The same also happens with Snort and Profiling detectors located on the Firewall/Gateway computer.

8.4.3. SMTP Flood

The next attack scenario is that of an e-mail spam flood towards an external SMTP Server. This experiment represents a hypothetic scenario where corrupted computers by Storm worm [22][86] are trying to infect other hosts via e-mail.

The Experiment

In order to achieve this setup, four (4) computers (all of them running Linux distribution) are used to execute a simple shell script (called spamflood.sh) that uses Packit tool to initiate TCP communication (TCP SYN) with GPRT SMTP server. Having duration of five minutes, the script uses forged IP addresses (and subnets) with randomly allocated client port numbers in the originator's addressing fields.

Figure 8.11 illustrates the number of packets destined to GPRT's SMTP server during 15 minutes, between 09:15 and 09:30 (the time of highest activity from GPRT users) of June, 09 2010. It is clear that the amount of packets increase after the attack is started. Before the attack taking place, the mean of SMTP packets was around 305 per minute whereas it suddenly increased to approximately 18000 packets per minute once the attack was launched. It is important to explain that this specific time interval was chosen because it represents the period with increased SMTP activity by GPRT users.

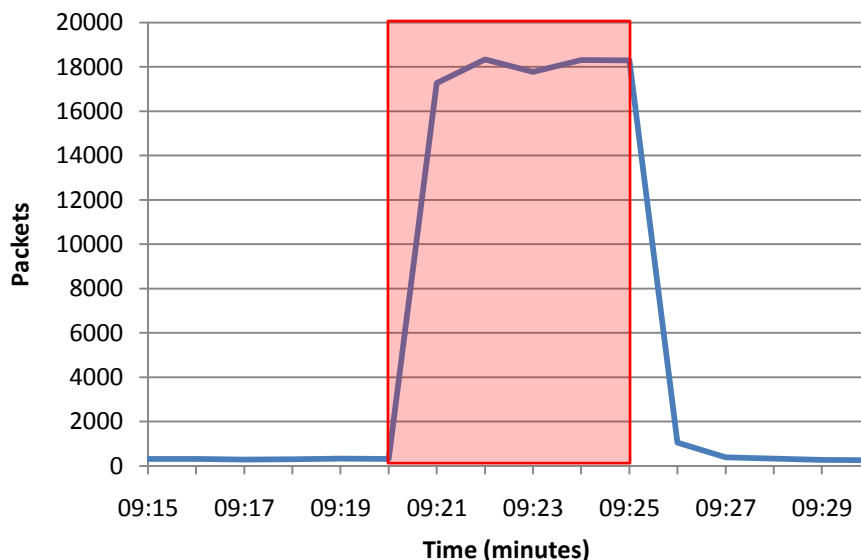


FIGURE 8.11: SPAM attack without defense (collected from Firewall/Gateway)

Analysis

Before the attack was started, none of the four detectors acting in this experiment (see Table 8.3) reported any alert, but this changed soon after the first minute of attack.

Approximately 5 seconds into the second minute (09:21), the Alert Pre-Processor received a single alert file from the ChkModel detector summarizing events that took place during the first minute. It contained 1160 alerts and indicated **SUSPICIOUS** activities from different hosts towards GPRT's SMTP server (150.161.192.192). Since the detection mechanism of ChkModel is based on the ratio of sent and received packets, all these are marked as suspicious because there is a rate of 1 to 0 observed. In other words, one packet is sent while none were received back.

Regarding the orchestration analysis, the significant cluster extraction process confirmed that all alerts were significant (using destination address as a key) and, for this reason, they were forwarded to the Decision Service. Here the state machine was used to evaluate them. Considering that the lack of Profiling and Snort alerts is represented by *good* state, the result of the state machine analysis is the state **B1** (represented by the combination: Profiling=*good*, ChkModel=*suspicious* and Snort=*good*). Consequently, this decision was translated to the following actions:

- To limit during 60 seconds any packet destined to IP address 150.161.192.192 with destination port 25.

Figure 8.12 shows an example of how such decision may be enforced at the firewall level.

```
IPTABLES -A INPUT -s 192.168.0.5 -d 150.161.192.192 --dport 25 -p tcp -m limit --limit 60/second -j DROP
IPTABLES -A INPUT -s 192.168.0.1 -d 150.161.192.192 --dport 25 -p tcp -m limit --limit 60/second -j DROP
IPTABLES -A INPUT -s 192.168.10.11 -d 150.161.192.192 --dport 25 -p tcp -m limit --limit 60/second -j DROP
IPTABLES -A INPUT -s 192.168.243.13 -d 150.161.192.192 --dport 25 -p tcp -m limit --limit 60/second -j DROP
.....
```

FIGURE 8.12: Possible IPTables rules for SMTP flood attack

In the next minute (09:22) a change of roles happens. The ChkModel detector, which had classified the initial e-mail traffic as suspicious, adjusts its thresholds and from now on it considered all traffic as legitimate since the TCP handshake mechanism for connections establishment is correctly used. For this reason, it not generates any alert. On the other hand, Profiling generates 1 alert file (containing 1240 source IP addresses) about this “massive” anomaly and sends it to the Alert Pre-Processor.

As the Profiling alert is composed by a unique IDMEF alert, it was not considered significant by the cluster extraction and also could not be applied in FER analysis as it did not fulfill the minimal count requirement (number of alerts > *fer_threshold*). Thus, it was sent to DST analysis. However, it was also refused because has a unique source. Lastly, the only possible solution is sending it to the Decision Service.

After evaluate the Profiling alert, Decision Service, using the state machine analysis, attributes the state **C1** (combination of Profiling=*bad*, ChkModel=*good* and Snort=*good*). Consequently, this decision was translated to the following actions:

- To limit during 300 seconds any packet destined to IP address 150.161.192.192 with destination port 25.

It is important to observer that since the Profiling was developed to detect massive anomalies, its alerts have more weight in the state machine analysis.

For the next minutes (09:23, 09:24 and 09:25), until the end of the attack, only the Profiling detector continues to generate alerts (containing 1055, 890 and 335 source IP addresses, respectively) and send them to be analyzed. The evaluations using state machine are the same (state **C1**) and the decisions taken are also similar: to limit during 300 seconds any packet destined to IP address 150.161.192.192 at port 25.

Figure 8.13 shows a complete view of this spam attack and its detection. It is straight forward to see that only the Profiling strategy was capable of detecting the attack. This can be explained by the fact that for the ChkModel, spam e-mail is legitimate traffic as it looks just like normal e-mail when making use of the TCP handshake mechanism for connection establishment. In other words, Spam email does not violate the TCP model of connection establishment. It is its content that is harmful and wasteful of user time. On the other hand, the Profiling detects the attack as it senses a sudden increase of the number of flows targeting a single SMTP server or IP address. However, the response time for the Profiling remains relatively high as it borders the two minutes.

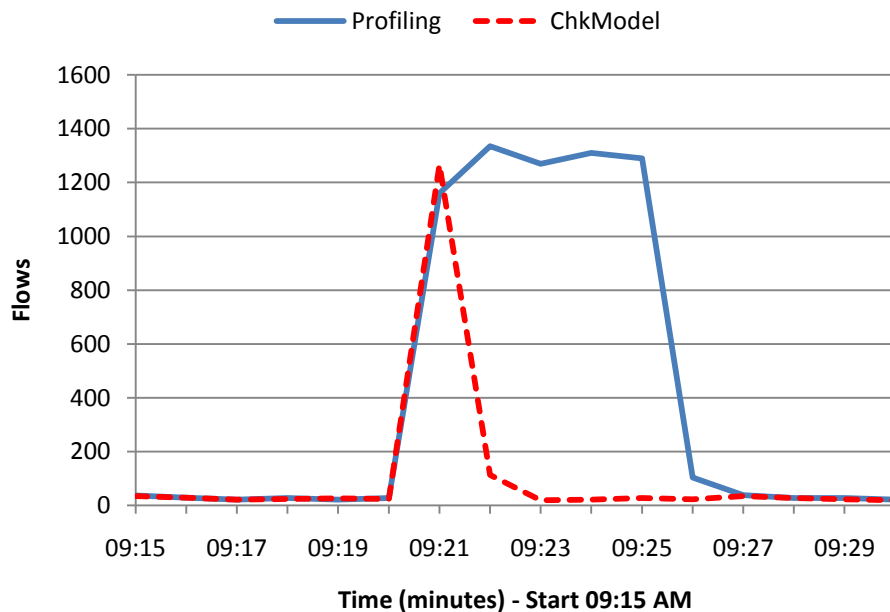


FIGURE 8.13: SPAM attack with defense

It is important to explain that Figure 7.13 is using flow information as parameters because both detectors (ChkModel and Profiling) employ this type of aggregation to perform their analysis and evaluations.

To sum up, the decisions taken by OADS (via its own Decision Service) in this experiment can be considered correct and functional. However, it is undeniable that they are also somewhat inefficient. The actions of limiting all and any packet targeting to SMTP server (150.161.192.192) with destination port 25, although proven to be effective, also do stop legitimate and correct connections of GPRT users. A good solution for this issue can be the use of Trusted IP List (TIL) [173][237]. The main idea of TIL consists in keeping a table with the description of the history of “good” connections already established with the network, so that during attack situations such as these are favored with most of the bandwidth available to the detriment of unknown connections and/or possible aggressors who will be limited by filters. Traffic shaping may therefore be used to differentiate both types of TCP connections.

8.4.4. Slowloris

Slowloris [235] is a low rate service denial attack (though it really not is a DoS attack). It operates by sending legitimate but incomplete HTTP requests, very similar to SYN

flood packets, but at a higher layer (the application layer in this case). This results in fewer packets needed and more granularities to collapse a Web server.

Slowloris attack takes advantage of Web server design, typically protected of massive attacks (mainly DDoS), occupying all available sockets, making that the server “waiting for the resting of requests”. Figure 8.14 illustrates a Slowloris HTTP request.

```
POST /somepage.com HTTP/1.1 r n
HOST: some_url_or_other.com r n
User-Agent: Mozilla/4.0 r n
Content-Length: 42 r n
X-a: b r n
```

FIGURE 8.14: Typical slowloris HTTP request

What differentiates the example of a functional HTTP request is the final line, where it should be finished off with an additional *r n* (Carriage Return and New Line characters). So the last line should be: *X-a: b r n r n*. This simple lack of *r n* causes some web servers to wait for completion, which is not unreasonable: maybe the missing carriage return/line feed (CRLF) is still on its way. Waiting is also one way of protecting the server against a brute force attack such as DDoS. The problem is that, by default, some Web servers will wait five minutes. That is, as a result, there is one resource that is used up for five minutes, unnecessarily in this case of Slowloris. However, it is important that each resource is kept busy, so every so often a new header is sent with the missing CRLF. If the exact form of this header is each time changed then it makes writing intrusion detection signatures harder.

The Experiment

In order to execute this experiment, two computers, one located in Informatics Center (Cin) of the Federal University of Pernambuco (UFPE) and the other one located in the Science Computer Department (DCC) of the Federal University of Amazonas (UFAM) were used to attack GPRT Web servers as depicted in Figure 8.3. These attackers were set up to shoot simultaneous attacks to the targets. In addition, in order to observe the attacks and their effects, a vulnerable distribution of the Apache Web server on the two targets was installed. However, it was decided to only enforce the decisions to stop the attack to the Web server 1 (150.161.192.192), as it is a production server.

It is important to explain that although the Slowloris script uses random source IP address, the attackers were located at networks behind NAT servers. For this reason, only two distinct source IP address will be perceived in this attack.

The experiment was initiated at 15:00 PM of June 17, 2010, during 20 minutes. Figure 8.15 illustrates the increase of the number of established connections destined to the GPRT Web server 2 (150.161.192.51), where clearly it is possible to see after the attack taking place, the Web server reaches its maximum capacity of connections (150).

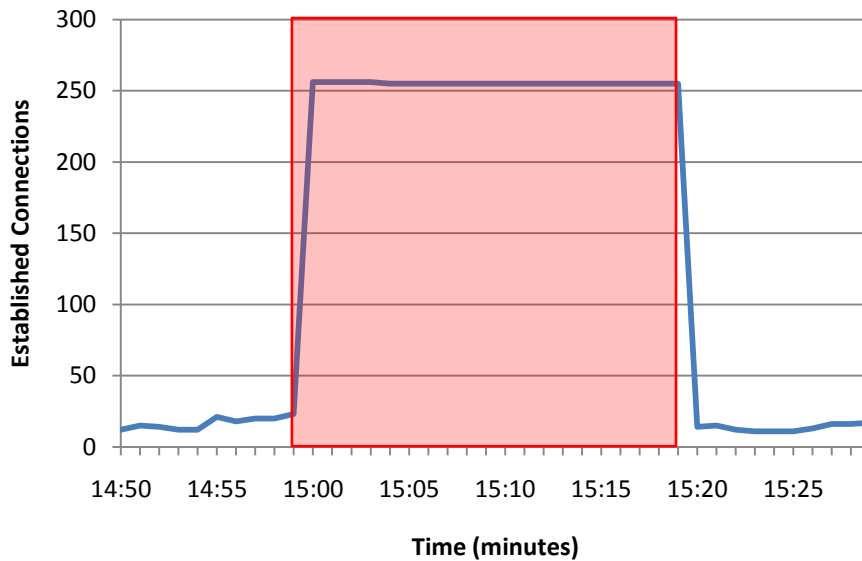


FIGURE 8.15: Slowloris attack without defense on Web server 2

Analysis

The Slowloris attack took place at 15:00. Two minutes into the attack (15:01), among the seven (7) detectors used in this scenario, Profiling and Snort using emergent configuration (Emergent threats) were unable to generate alerts. The explanation for this is simple. As this attack has the same behavior of a low rate SYN flood, it generates a low number of flows and hence slips through the control of the Profiling technique. Snort no built-in rule capable of classifying this attack.

Nonetheless, both Snort's and ChkModel detectors saw a number of "TCP SYN" packets going through towards the same destination servers and therefore should be capable of detecting the attack. In fact, the Alert Pre-Processor received 7 IDMEF alerts file (4 for Web server 1 and 3 for Web server 2). Snort classifies the attack as "*SPECIFIC-THREATS Slowloris http DoS tool*" with severity equal to 2, whereas ChkModel classify the attack as BAD.

Please recall that according to Algorithm 8.1, the execution of significant cluster extraction is the first step. However, at the end of such process, all five alerts were considered not significant since they were only few of them and had little differences among them. Hence, the next step is to verify if these alerts can be evaluated by the FER Analyzer. Again, due to their reduced number, the alert also failed to pass this verification. Consequently, they are then sent to the ADS-Fusion (step 5). Unlike other attacks, this one generated alerts from distinct detector, which make them good candidates to evaluation using the Dempster-Shafer Theory.

As explained in section 8.2.2, the ADS-Fusion begins with the synchronization of alerts aiming to establish connection among them. So, the received alerts are aggregated according to their affinities. In other words, the alerts are split according to their targeting of Web server 1 or Web server 2. After that, they are combined to generate *bpa* values. For this, their severity parameter, the number of equal alerts and thresholds are used. The two first are extracted from Snort alert and the latter from ChkModel alerts. Then, the *bpa* for each alert is calculated. Snort alerts (Firewall/Router and Web server 1) have only a single alert each with severity as medium. This way, the calculated *bpa* for each one is 0.65 (what correspond to medium severity as explained in section 8.2.2). For the ChkModel alerts (Firewall/Router and

Web server 1), the calculated *bpa* is based on threshold values. So, the calculated *bpa* for each one is 0.8. Tables 8.4 and 8.5 describe the *bpa*'s of all alerts.

TABLE 8.4: Calculated *bpa*'s of Web server 1

Detector	Severity	Threshold	Bpa
<i>Snort (Firewall/Router)</i>	medium	0.65	0.65
<i>ChkModel (Firewall/Router)</i>	bad	0.8	0.8
<i>Snort (Web server 1)</i>	medium	0.65	0.65
<i>ChkModel (Web server 1)</i>	bad	0.8	0.8

TABLE 8.5: Calculated *bpa*'s of Web server 2

Detector	Severity	Threshold	Bpa
<i>Snort (Firewall/Router)</i>	medium	0.65	0.65
<i>ChkModel (Firewall/Router)</i>	bad	0.8	0.8
<i>ChkModel (Web server 2)</i>	bad	0.8	0.8

The last step is inferences generation. This step requires the definition of a frame of discernment, an element that contains the possible states of the network, and the hypothesis being evaluated. In this work, all generated frames of discernment have only two possible elements to represent the network state: Normal or Anomalous ($\Theta = \{Normal, Anomalous\}$). In addition, the hypothesis to be questioned always is if the network state is Anomalous, that is, the network is under attack ($H = \{Anomalous\}$).

Next, the belief, $bel()$, and plausibility, $pl()$, functions are calculated, considering the hypothesis H , and, as consequence, a range of belief, $\mathcal{J}(H)$, which expresses the range of values in which it is possible to believe in the hypothesis H , is generated.

Overall, ADS-Fusion considers for Web server 1 that:

- m_1 as the mass function of the Snort attack evidence from Firewall/Router, m_2 as the mass function of ChkModel attack evidence from Firewall/Router, m_3 as the mass function of Snort attack evidence from Web server 1 and m_4 as the mass function of ChkModel attack evidence from Web server 1.
- Frame of discernment is $\Theta = \{Normal, Anomalous\}$, and:
 - $m_1(Anomalous) = 0.65$ and $m_1(\Theta) = 1 - m_1(Anomalous) = 0.35$
 - $m_2(Anomalous) = 0.8$ and $m_2(\Theta) = 1 - m_2(Anomalous) = 0.20$
 - $m_3(Anomalous) = 0.65$ and $m_3(\Theta) = 1 - m_3(Anomalous) = 0.35$
 - $m_4(Anomalous) = 0.8$ and $m_4(\Theta) = 1 - m_4(Anomalous) = 0.20$
- The belief and plausibility for all mass functions are 1, i.e., $bel(\{Normal, Anomalous\}) = 1$ and $pl(\{Normal, Anomalous\}) = 1$.

As result, the Dempster combination obtained the following values: $m_1 + m_2 + m_3 + m_4(\{Anomalous\}) = 0.881355$ and $m_1 + m_2 + m_3 + m_4(\{Normal\}) = 0.118645$.

For Web server 2, ADS-Fusion considers that:

- m_1 as the mass function of Snort attack evidence from Firewall/Router, m_2 as the mass function of ChkModel attack evidence from

Firewall/Router, m_3 as the mass function of ChkModel attack evidence from Web server 2.

- Frame of discernment is $\Theta = \{ Normal, Anomalous \}$, and:
 - $m_1(Anomalous) = 0.65$ and $m_1(\Theta) = 1 - m_1(Anomalous) = 0.35$
 - $m_2(Anomalous) = 0.8$ and $m_2(\Theta) = 1 - m_2(Anomalous) = 0.20$
 - $m_3(Anomalous) = 0.8$ and $m_3(\Theta) = 1 - m_3(Anomalous) = 0.20$
- The belief and plausibility for all mass functions are 1, i.e., $bel(\{Normal, Anomalous\}) = 1$ and $pl(\{Normal, Anomalous\}) = 1$.

As result of the Dempster combination, the same values of Web server 1, i.e., $m_1 + m_2 + m_3(\{ Anomalous\}) = 0.881355$ and $m_1 + m_2 + m_3(\{ Normal\}) = 0.118645$, were obtained.

Since the inferences were generated, according to Algorithm 8.1, the next step is sending the results to OADS's Decision Service. Note that the results include the inferences and original alerts.

After a validation, the Decision Service takes the following decisions:

- To block all packets sent by IP addresses 150.161.2.53 (CIn/UFPE) and 200.17.49.5 (DCC/UFAM), destined to Web server (150.161.192.192) on port 80.

Consequently, the Slowloris attack to Web server 1 is blocked and the detectors do no generate any more alerts. Figure 8.16 demonstrates the attack effects on Web server 1. Note that after the first minute of the attack, the enforcement actions are taken and the attack is blocked.

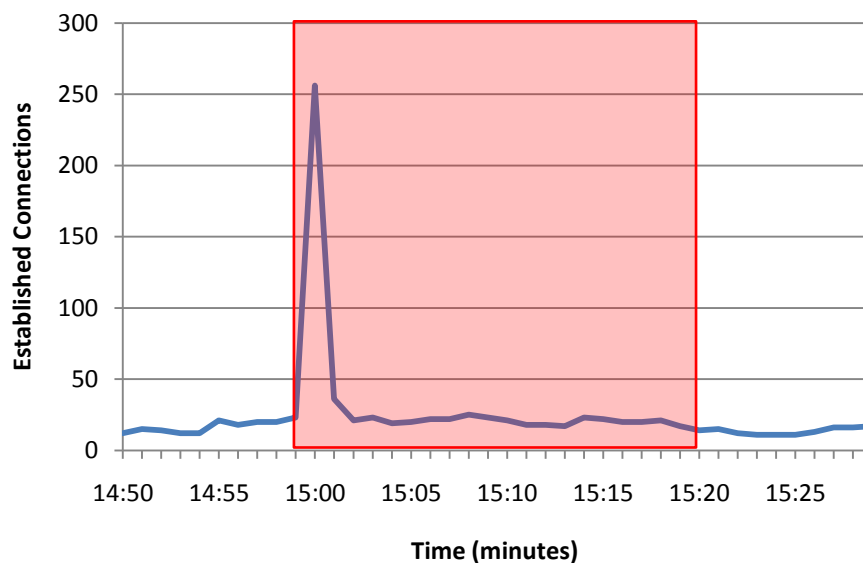


FIGURE 8.16: Slowloris attack without defense on Web server 2

For effect of evaluation, it was decided not to apply the enforcement actions destined to Web server 2. In other words, the decisions of blocking the packets of the attackers are not configured, hence allowing the monitoring of the subsequent alerts and their analysis by ADS-Fusion. Table 8.6 represents the individual belief of the detectors and the Dempster combination for the hypothesis of the network is under attack.

TABLE 8.6: Dempster combination for attack in Web server 2

Time	Snort (Firewall/Router)	ChkModel (Firewall/Router)	ChkModel (Web server 2)	DST
15:00	0.65	0.80	0.80	88%
15:01	0.70	0.80	0.80	90%
15:02	0.75	0.80	0.80	92%
15:03	0.80	0.80	0.80	94%
15:04	0.85	0.80	0.80	95%
15:05	0.90	0.80	0.80	97%
15:06	0.95	0.80	0.80	98%
15:07	1.00	0.80	0.80	99%
15:08	1.00	0.80	0.80	99%
...				
15:19	1.00	0.80	0.80	99%

Note that the belief of the Snort detector increases as the number of repeated alerts is received. For example, during the initial time period (15:00), the unique alert has its belief equal to 0.65. At the second time (15:01), one more equal alert was received. For this reason, its belief was 0.65 (its severity) + 0.05 (second apparition of the same alert), totalizing 0.7. As a result, the Dempster combination for the network is under attack hypothesis also increases. At 15:07, after receives 8 identical alerts from Snort, the belief achieved the maximum level (1.0). From this point on, ADS-Fusion achieves almost 100% of belief (confidence) regarding the existence of an attack.

8.4.5. Multi-step Attack

The last experiment (but not the least important one) is composed of a set of attack actions, that is, a multi-step attack. According to Robiah *et al.* [238], a multi-step attack is a sequence of attack steps that an attacker has performed, where each step of the attack is dependent on the successful completion of the previous one.

The interesting and relevant aspect of multi-step attacks is that they can and must be observed by different detectors. However, it is necessary to gather all pieces so that an attack scenario can be seen a multi-step attack.

For the experiment at hand, a multi-step attack scenarios caused by the Blaster worm [239] spreading mechanism was emulated. Blaster worm scans the local class C subnet, or other random subnets, on port 135, in an attempt to discover vulnerable systems and thus use them as targets. The exploit code opens a backdoor on TCP port 4444 and instructs them to download and execute the file MSBLAST.EXE from a remote system via the Trivial File Transfer Protocol (TFTP) running over UDP port 69 to the `%WinDir%\system32` directory of the infected system [238].

The Experiment

In order to implement a Blaster worm experiment, a different testbed, shown in Figure 8.17, was built. As one is dealing with an internal attack, during 30 minutes, as many as 60 computers of GPRT laboratory, where nine (9) computers, running Linux distribution, were “prepared” to makeup this experiment. One of these was selected to act as the attacker machine whereas the other emulated Windows machines.

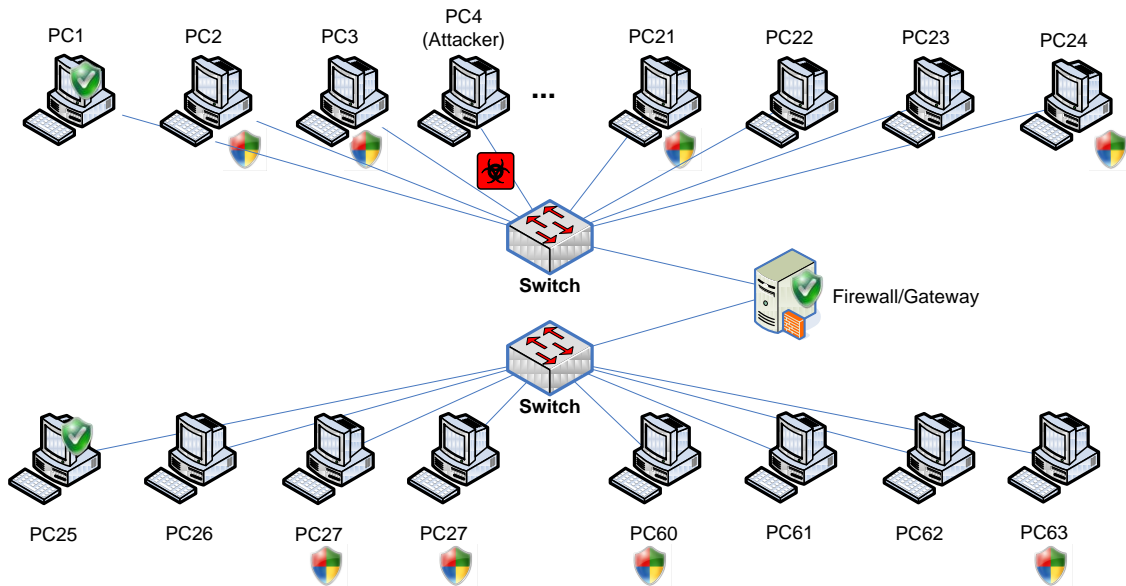


FIGURE 8.17: Blaster worm testbed scenario

Based on the Blaster worm operational steps described above, the experiment had the following behavior. First, the attacker was activated and it began a scanning process in the network, looking for open port 135 TCP to explore DCOM RPC vulnerability in Microsoft Windows. For this, a port scan script (pscan.py) with 192.168.0.0/24 as target and 192.168.0.96 as source IP address was used.

The second step consisted in exploring the vulnerability on TCP port 135. In order to emulate this step, the attacker executed a script (blaster.py) whereas each one of the 8 “vulnerable” computers executed other script (blaster_client.py) to communicate with the attacker. Basically, the attacker sent a message instructing the vulnerable computers to open a backdoor on port 4444 TCP. As a proof of concept, the attacker script tried to next to connect to the vulnerable computers on port 4444 TCP and get an image file, called BLASTER.jpg. A specific Snort rule to detect this communication, as shown in Figure 8.18, was developed.

```
alert tcp $HOME_NET any -> $HOME_NET 4444 (msg:"Blaster Worm Simulation 4444";
flow:established; uricontent:"BLASTER.jpg"; nocase; sid:1000001; rev:1);
```

FIGURE 8.18: Specific Snort rule to detect Blaster worm simulation

At a third step, the vulnerable computer makes a TFTP connection to the attacker, to get the file MSBLAST.EXE. In order to turn viable this step, the TFTP API [240] was used to implement all TFTP communication between attacker and vulnerable computers. After this step, the attacker closes its activities.

To finish the attack, all vulnerable computers tried to establish connections with Web sites where this Storm worm [22][86] can be found, in order to create a new infection. For this, each one of them has a list containing 10 Web site addresses recognizably related with this worm, and chooses only 2 to try a connection. After this step, as the attacker, the vulnerable computers also close their activities.

It is interesting to observe that multi-step attack scenarios must be observed by different detectors, like signature-based network IDS, ADS and file integrity checker. However, in this scenario only Snort detectors were used due to the fact that they are more prepared, regarding the existence of rules. In addition, both ChkModel as Profiling are not adequate in this case, since this experiment does not generate differences

between ingress and egress TCP packet (what discards the use of ChkModel) nether does it generate huge amount of traffic, hence discarding the use of Profiling to detect it. For such reasons, only Snort detector versions 2.8.3.2 and 2.8.6 are used.

Analysis

The Blaster worm experiment was started at 10:00 of July 26 2010. After the first minute, the Alert Pre-Processor received three alert files from Snort Firewall/Gateway (version 2.8.6) and Snort's PC 1 and PC 25 (version 2.8.3.2), containing 240, 193, and 194 alerts, respectively.

These alerts represent all attack steps and were classified by Snort's as "PSNG_TCP_Portsweep" (551 alerts), "Blaster worm simulation 4444" (24 alerts), "TFTP Get" (24 alerts), and "Storm worm phone address" (48 alerts). The first represents the port scan activity. The second one represents TCP connection opening to port 4444 in vulnerable computers. The third represents the TFTP connection to get MBLASTER.exe file. The last alerts show the attempt to connect to Web sites related to the Storm worm.

With regard to the orchestration analysis (according to Algorithm 8.1), the alerts are evaluated using the significant cluster extraction process. As a result, all alerts of "PSNG_TCP_PORTSWEEP" type are considered relevant ($RU(A) = 0.5074532071$ of class and $RU(A) = 0.8054653980$ of *srcIP*). Next, they are forwarded to the Decision Service, which decides to block any packet sent by IP address 192.168.0.96, destined to 192.168.0.0/24, as shown in this simple IPTables example (`iptables -A INPUT -s 192.168.0.96 -d 192.168.0.0/24 -j DROP`).

One must emphasize that although an enforcement action is taken by the Decision Service, it has no effect. The reason is simple. In this testbed, the attack is totally inside the network (internal to the network) making the first point of enforcement, namely, the Firewall/Gateway computer useless in such case. This way, although an explicit order was issued to block all packets from this source, their presence continues in the network.

Nonetheless, there are still other alerts to be analyzed. Those alerts that were considered as irrelevant at step 2 of the orchestration algorithm may still be used. The next step is to verify if these alerts can be evaluated by the FER analyzer. As the number of alerts is greater than *fer_threshold* ($96 > 50$), the discovery of frequent episodes is applied.

Consequently, all alerts are translated into events. Table 8.7 exemplifies some alerts and event types in this scenario.

TABLE 8.7: Example of event types and event names for Blaster worm scenario

Event Type	Event Name	Source IP/Port	Destination IP/Port
A	Blaster worm simulation 4444	192.168.0.96:34521	192.168.0.50:4444
B	Blaster worm simulation 4444	192.168.0.96:50674	192.168.0.51:4444
H	Blaster worm simulation 4444	192.168.0.96:12543	192.168.0.57:4444
I	TFTP Get	192.168.0.50: 5643	192.168.0.96:69
J	TFTP Get	192.168.0.51: 3027	192.168.0.96:69

P	<i>TFTP Get</i>	192.168.0.57: 3027	192.168.0.96:69
Q	Storm worm phone address	192.168.0.50: 64267	222.252.232.184:22861
R	Storm worm phone address	192.168.0.50: 4530	216.139.142.17:10788
F1	Storm worm phone address	192.168.0.57: 1155	217.77.54.253:12358
G1	Storm worm phone address	192.168.0.57: 7897	222.33.177.224:12555

Next, using the established parameters from Table 8.1 (window size 10, frequency threshold 0.01, and confidence threshold 0.8), the computation of frequent episodes is made and the following values (Table 8.8) are discovered.

TABLE 8.8: Performance for Blaster worm scenario

Window Size	Candidates	Frequent Episodes	Level of Participation (%)
1	32	32	100.00%
2	289	153	52.94%
3	1376	612	44.47%
4	3468	1428	41.17%
5	5712	2142	37.50%
6	6426	2142	33.33%
7	4998	1428	28.57%
8	2652	612	23.07%
9	918	153	16.66%
10	187	32	17.11%

Note that the presence of a low number of alerts generating a more focused number of frequent episodes. Such affirmation is proved by the final number of frequent episodes found (32) for a maximum value window size.

The next step of FER analysis has to do with episode rules generation. FER generates 4334 normal rules and 137 reduced rules, respectively, using frequency threshold of 0.01 and confidence level of 0.8 (Table 8.1). Among the reduced rules, it is possible to find representations of the multi-step attack. For instance, the rules $A \rightarrow AI$ with confidence 1.00 and $I \rightarrow IQR$ with confidence 1.00 allow deducting that in 100% of the cases of event A (Blaster worm simulation on port 4444, from 192.168.0.96:34521 to 192.168.0.50:4444) occurs, the event sequence $AIQR$ (Blaster worm, TFTP Get, Storm worm and Storm worm) also occurs.

The processing time including frequent episodes calculation and episodes rule generation was around 67 seconds.

Since the episodes rules were generated, the next step is sending these rules (and event tables) to the Decision Service. After validation, a series of decisions is taken to block the communication between the attacker (192.168.0.96) and the vulnerable computers (192.168.0.50-57). Note that these decisions have no effect. Once again, these decisions will be applied in Firewall/Gateway computer, but all communication between attacker and vulnerable computers actually do not pass through this network point of ingress/egress traffic.

As such, all this described process is repeated until the attack scripts stop working. A solution would be achieved through the use of an automatic access control mechanism as proposed in [241], where the authors employ 802.1x [242] to implement access control based on physical access device ports. This way, specific user traffic may be filtered out at the access switches.

8.4.6. Experimenting with real traces

In order to evaluate the OADS approach, some actual traffic traces were used.

The first one was CAIDA's 2007 DDoS Attack Dataset [243], obtained from CAIDA (Cooperative Association for Internet Data Analysis [139]). This trace contains approximately one hour of anonymized traffic traces from a DDoS attack that took place on August 4th, 2007 (20:50:08 UTC to 21:56:16 UTC). This type of denial-of-service attack attempts to block access to the targeted server by consuming computing resources on the server and by consuming all of the bandwidth of the network connecting the server to the Internet. The total size of the dataset is 21 GB, where only attack traffic to the victim and responses to the attack from the victim are included.

Although this trace contains bidirectional traffic, none of the detectors used in testbed topology was able to identify any attack. The reason is simple. Since the payload was removed from all packets in this trace, Snort detectors would hardly find any signature. Regarding ChkModel and Profiling, there was no satisfactory explanation of why they did not detect anything. There is the possibility that the anonymization process may have affected the trace data.

The second trace is CAIDA's Backscatter 2008 Datasets [244]. It consists of quarterly week-long collections of responses to spoofed traffic sent by denial-of-service attack victims and received by the UCSD Network Telescope. Data was collected quarterly in February, May, August and November. In addition to the quarterly collections, data was also collected on March 18 and 19 for the Day in The Life of the Internet (DITL) project. Only this last one was experimented with.

Since this trace is a backscatter from DoS victims (responses), only the Profiling detector can use them to look for anomalies (the ChkModel needs requests and responses to perform analysis and the Snort needs a payload). This way, it was decided not to use this trace since, according to Algorithm 8.1, the final result will be getting Profiling alerts (one each five minutes) and simply forwarding these to the Decision Service.

The third and fourth traces are UMass Gateway Link 3 Trace, obtained from UMass Trace Repository [245], and MAWI 2006 samplepoint B, obtained from MAWI Working Group Traffic Archive [246]. The former is a collection of traces taken at the UMASS OIT gateway router. The traces are collected every morning from 9:30 to 10:30 from July 9th, 2004 to July 22nd, 2004. They are in DAG format. All the IP addresses have been anonymized with prefix preserving algorithms. The latter is collected from trans-Pacific line (18Mbps CAR on 100Mbps link) during the six first months of 2006. The traces of June 1st, 2006 were used as discussed next.

Regarding these two traces, both were injecting into the OADS prototype (separately), but none of them generates any type of alert. The MAWI trace not indicates if there is or not attacks and anomalies in its data. On the other hand, UMass trace is listed as containing anomalous traffic. But a simple evaluation using

TCPDump³¹ and WireShark³² tools show that there is not unwanted traffic in its data, the reason because nothing was found.

8.5. Chapter Discussion

This chapter has presented and described the implementation and evaluation of the OADS approach conducted as part of this study to show the effectiveness of the proposed ideas. A key feature of the implementation was the use of a simple heuristic (represented by Algorithm 8.1) instead of more complex solutions, allowing that OADS approach be more agile and flexible.

In order to evaluate the OASD prototype, different experiments in a controllable environment (GPRT laboratory) were conducted, proving that the proposal is very capable to identify and stop unwanted traffic. Unfortunately, the many attempts to execute OADS approach with public and real traffic traces were not well successful. Basically, the presence of backscatter traffic (only the traffic from the victim(s) to the attacker(s) is recorded) restraints that the used detectors discovery some kind of attack or anomaly.

Although early, it is fair to state that the approach may be seen as a significant step towards building sound security and unwanted traffic detection and mitigation unified platforms.

³¹ <http://www.tcpdump.org>

³² <http://www.wireshark.com>

Chapter 9

Conclusions

A key contribution of this Thesis is to allow that network operators and IT managers be free of everyday and cumbersome tasks of evaluating security events, alerts, and incidents provided by the numerous network security software and services. For this goal, a generic, open and flexible approach to deal with unwanted Internet traffic was designed and implemented. It is generic, because its components may be organized to detect a large and varied range of intrusions, attacks and anomalies, spanning from a traditional LAN access control service to large high-speed ISPs and backbones. Its openness stems from employing open source languages, standard protocols, and tools. It is flexible in that allows the update or seamless insertion of new detection techniques.

Our approach is a holistic view of the orchestration concept applied in intrusion and anomaly detection. Named Orchestration Anomaly Detection System (OADS), it offers support for collaboration and harmonization of different detectors, increasing the power of perception (detection) of anomalies and consequently turning the network more and more secure. In practical terms, OADS is a complete unified framework for unwanted traffic identification able to deal with different detectors and their multitude of alerts, employing different techniques and methods for analysis to confirm or deny the presence of intrusions, attacks and anomalies and indicate some type of enforcement action.

The rest of this chapter is organized as follows. We begin our conclusions by summarizing the main contributions of this Thesis. Then, we take a look back and report on the key lessons learnt in this research. Next, we show some perspectives, contexts and points to directions for future works. Finally, we provide concluding remarks.

9.1. Summary of Contributions

In this Thesis, we have advocated that the collaboration and harmonization among different anomaly detectors is a good step to achieve a desired result in security area. Motivated by this view, we proceeded to design and develop a complete unified framework for unwanted Internet traffic. We then applied this framework in different scenarios, aiming to validate it and consequently point out where and how it could be improved.

In particular, this Thesis makes the following contributions:

- **Unwanted Internet Traffic Survey:** In order to better understand the issues involving this traffic, we made an ample review of the subject. Our research began by surveying the unwanted Internet traffic (Chapter 2). We presented different definitions and formulated a new one that summarizes the other and includes aspects as the characterization of legitimate and illegitimate traffic as unwanted. It also discussed about context, classifications, and the possible reasons to explain its increase on recent years, followed by the presentation of many examples of unwanted traffic. Next, we made a survey

contemplating the most varied solutions employed against this traffic (Chapter 1). It includes strategies ranging from as simple as the used of anti-virus software to more advanced research tools for anomaly traffic detection.

- **Orchestration Approach for Unwanted Internet Traffic Identification.** The finding that collaborative unified solutions are needed to efficiently deal with the problem of unwanted traffic was instrumental in the direction of the adopted solutions. Our second contribution is the proposal of a new approach based on the orchestration of different security components for unwanted traffic identification (Chapter 4). In practical terms, the Orchestration Anomaly Detection System (OADS) specifies a unified framework capable of receiving multiples inputs (alerts) from different anomaly detectors, evaluating them and warning about possible problems. Enforcement actions can be taken, but they are not considered at depth by this Thesis.
- **OADS Miner (ARAPONGA).** Information is the key to any efficient solution. As such, network administrators and IT managers must rely on Web sites to get this information. Our third contribution is a tool capable of gathering information about vulnerability reports, security events and Internet traffic statistics, consolidating and indexing them into a single place. The result is a practical, simple, fast, useful and straightforward information source for users (Chapter 5). ARAPONGA contributes by applying the concepts of WIRSS and ISSS to provide functionalities support that go beyond a traditional system of searching and indexing. In addition, it uses data mining techniques and templates to expand the capability of indexing information about security and therefore allow different and more focused queries.
- **OADS Alert Pre-Processor tool.** In order to deal with the multitude of alerts from different detector, our fourth contribution is a solution for the aggregation and extraction of significant alerts (Chapter 6). Based on the clustering approach, the Alert Pre-Processor tool receives multi-source alerts, aggregates them and extracts the most relevant ones. Furthermore, in order to reduce the computational load at the (centralized) server and decrease the false negative rate, it can also be used in learning about attack strategies.
- **OADS FER Analyzer.** Our fifth contribution is in the form of a statistical module based on the frequent episodes discovery technique capable of correlating alerts, discovering sequences of events that represent strategies or phases of attacks and enabling the prediction of future alerts (Chapter 7). Moreover, as part of its analysis result, this tool generates probabilistic rules that can be used in the enforcement actions.

9.2. Lessons Learned

We now draw out some lessons that we learned through the development of the OADS approach.

9.2.1. Content Selection vs. Crawler Tool (OADS Miner)

There was a high level of difficulty encountered with the implementation of ‘OADS Miner. In particular, there was a difficult decision to be taken when choosing between

obtain all the information possible versus respecting the limits set by the visited Web sites for Crawlers. As previously mentioned (Section 7.3.1), we choose dozens of Web sites as point of gathering reliable security information and has to sometimes ignore the limits imposed by these sites for the purpose of building a useful database of events.

Unfortunately, around a dozens of important security information sites were discarded due to their limiting access policies to the content. More specifically, two issues were registered. The first one is related to robots.txt, a file found at the root page of each domain and created to control the actions of search robots, dictating their search behavior. Typically, robots.txt files are generated to hide all content of robots. They have the following settings: *User-agent: ** and *disallow: /*, which restricts any agent (robot) from accessing any content inside their directory. The second issue is related to the META-TAGs, HTML reserved keywords (labels) that among other functions describe what contents a robot can see. A typical example of anti-crawling META-TAGS has the following format (`<meta name="robots" content="index,nofollow">`), where *index* and *nofollow* fields indicate that permission is given to index only the initial page of Web site.

This Thesis seeks to offer mechanisms for gathering information for the combat of attacks and anomalies. For this, we should be change Nutch configuration (good behavior policies) to ignore robots.txt and META-TAGS restrictions, modifying the source code of Nutch. By definition, a Crawler tool that does not respect these policies is known as *Malware Crawler*. We opted for not to follow this line of thinking. So, we tried to contact many Web sites, explaining the Thesis intentions and the need to obtain the advertized information.

Many Web contacted sites did not respond to our solicitations and consequently they were not used. Fortunately other, like Atlas [63], made available special accounts with full access for their data.

9.2.2. Detectors and IDMEF

Although we have presented some interesting unwanted traffic strategies in Chapter 2, it was a hard task, in terms of implementation and adequacy to the work, to define and test anomaly detectors.

Among all presented academic strategies, we could only use two: ChkModel [158] and Profiling [151]. The former because it was originally designed and developed by GPRT team (as part of undergraduate work) whereas the latter was implemented following the original specifications. Both are used in the RobustIP project [173]. Regarding the other academic strategies, when requested, the authors could not make available their code due to privacy issues in their projects. In addition, as described in Chapter 4 and 6, we had to develop a parser to convert the outputs of detector or sensors such the ChkModel and Profiling into the IDMEF format. This simple tool was written in Java.

Regarding commercial detectors, we tested three tools: Snort [163], Bro [164] and Prelude IDS [165]. Although all of them seem to be similar in their objectives, we choose Snort (version 2.8.3.2) due to its easier installation and simpler update rules. Another point in favor of Snort is the existence of the Snort-IDMEF plugin [178] is used to translate Snort logs into IDMEF alerts. However, after isolated tests, we discovered that: (i) Snort 2.8.3.2 does not support the most recent rules, due to

incompatibility of implementations; (ii) Snort-IDMEF plugin only works with Snort version 2.8.3.2.

As result, we opted for using Snort 2.8.6 (the most recent version) and use Java code that translates Snort logs to the IDMEF format.

9.2.3. Real traffic traces

Despite the OADS evaluation being performed using attack scenarios (originated from four experiments), further evaluations may be conducted to better prove the effectiveness of the approach.

An interesting, problematic and further point to be studied is the use of real traffic traces. Despite the increased interest in security, especially after the increased use of social networks, it is practically impossible to find traces that allow us consistently to test any solution. On the other hand, the strategy to use attack descriptions to recreate multi-step scenarios depends of what to exploit and demands an extra effort in terms of requirements.

9.3. Future Directions

This thesis introduced a unified approach for unwanted Internet traffic mitigation, composed by distinct components and designed to be modular. However, this approach can be further extended and applied in other contexts as well. We now explore some avenues for future research based on the contributions in this Thesis.

9.3.1. Distributed support and cooperation

The collaboration between detection systems is extremely necessary to increase, in both quality and quantity, the detection process of anomalies, suspicious events, and security incidents and, for this reason, it is an important aspect that must be studied. This collaboration must be based on information exchange (data and control) between local and remote detection systems. Standard message formats and protocols such as IDMEF [88], IODEF [89], and IDXP [90] can be used for this purpose. Recently, the use of description languages as WSDL (Web Service Description Language) [247] and OWL (Web Ontology Language) [248] has been practiced in the context of IDS collaboration.

9.3.2. Secure and trusty relationship

Another possible future work is related with some security and trust relationship requirements. The first one is data privacy. It aims to deal with the unwillingness, by the most different reasons, of the participant in to share security alerts and information about their domains and users. Some works have been proposed to address this issue. Lincoln *et al.* [249] proposes a set of sanitization techniques to obscure sensitive fields (IP addresses and data) and sensitive associations (the configuration and defense coverage of a network site). Xu and Ning [250] proposed the use of concept hierarchies to balance privacy requirements and the need for intrusion analysis. Already Gross *et al.* [251] proposed a privacy-preserving mechanism using Bloom filters for use in a CIDS. Other security aspect is the use of authentication and data integrity to prevent that wrong or forged information to be injected as part of the generated messages by the elements of CAIDSs. The works described in [252] and [253] use certificates to authenticate the messages and thus to guarantee the security of alerts and participants. In

addition, these approaches use a central certificate authority unit that can cause scalability problems. The proposal of Brandão *et al.* [92] describes a framework to integrate IDS, called IDS Composition, based on Web Service technology, where a security service is established using WS-Security standard [254], XML-Encryption [255], and XML-Signature [256] to deal with authentication and access control of elements and to exchange IDMEF messages.

9.3.3. Studies on distribution of the OADS orchestration

An interesting work would be to study the feasibility of distributing the orchestration functionality. Since the collected traffic can easily exceed the processing capacity of a server, especially in broadband networks, OADS orchestration can be quickly flooded by hundreds or thousands of alerts and diagnosis in a short time.

Intuitively, we can point out two solutions to solve the problem. The first one is to use more than one OADS orchestration engine to share the responsibility for the analysis and decisions. The second one (more interesting, but more complex) exploits the concept of parallelism to harmonize multiple OADS orchestration. This harmonization requires cooperation based on the use of fixed time intervals for making an analysis and the same dictionary of decisions.

9.3.4. Improvements in Orchestration algorithm

Other further work would be to study more attack scenarios to identify new and affective orchestration sequences. Although the proposed orchestration algorithm has been efficient, the current implementation may be seen as a proof of concept only. So, it is possible to design other scenarios and experiments, i.e., the current heuristic cannot be sufficiently prepared to detect all types of unwanted traffic. For this reason, extensive studies and tests are required to build orchestration sequences that are more effective and able to reduce the time taken to detect unwanted traffic.

9.3.5. Design and implementation of an inter-domain advertisement service

Another possible future work is the design and implementation of an advertisement service. In OADS approach, we advocated that the collaboration concept can and must be extended to external elements, systems or networks (Chapter 4), allowing that all participating domains of some type of collaboration must publically advertize the detection scope and types of defenses that they offer, opening opportunities for possible cooperation with others domains or similar systems.

The kickoff to address this issue begins by making available our information bases, including alert (received and pre-processed), analysis of OADS component (FER Analyzer and ADS-Fusion, for example), history of decisions and vulnerabilities reports and Internet traffic statistics (gathered by OADS Miner), in a shared storage space.

9.3.6. Information bases

In order to help on the anomaly detection process and attempt the advertisement scope requirement, we believe that the use of distinct information bases to store available and useful data about alerts and traffic summaries (anomaly detectors output), vulnerabilities and Internet anomalies, and history data is very interesting. The idea

behind the information bases is to offer for the orchestration approach a variety of knowledge about network state, allowing that it can take more satisfactory and correct decisions.

In OADS approach, for example, two data structure (information bases) to help on the orchestration tasks would be deployed. They are:

- **Alert base** contains the outputs of the analysis or traffic summaries performed by anomaly detectors. The idea is keeping processed network traffic while respecting its specificities. These information can used by OADS as important feedback for improving collaborative activities, taking decisions or looking for new anomaly patterns based on these observed results. For instance, the OADS approach can determine that a detector specialized in spam uses the generated alerts by the profiling [151] detector to evaluate further those results considered suspicious and not yet confirmed involving TCP port 25 (SMTP protocol). Such feedback procedure increases the chance of discovering unperceived and new anomalies and consequently enhancing the network security level.
- **History base** contains all previous historical decisions taken by the OADS approach for possible future processing. This way, when similar network situations happen, including the same network behavior and anomaly detectors being involved, the OADS approach can be compared or even take the same old decision.

9.4. Final Remarks

Unwanted Internet traffic can and must be considered a plague. Although the types and their consequences are known, issues such as the definition and ways to minimize its effects are still under discussion and study. The existence of an “evil industry” motivated and evolved, coupled with the emergence of new services and applications, the constant technological evolution and the population boom of new (and often unskilled) users imposes new challenges in the activity of detection and limitation of unwanted traffic.

This Thesis has made the case for studying unwanted Internet traffic and proposed an orchestration oriented anomaly detection system (OADS) approach for unwanted Internet traffic identification. It has also demonstrated the effectiveness of the OADS approach in the identification and mitigation of the unwanted Internet traffic through different scenarios and experiments.

To conclude this Thesis, it is notorious that the problem of the unwanted Internet traffic identification is still far from being solved, that one is dealing with an evolving problem and that only the tip of the iceberg was touched. However, there is also a general agreement both in academic as well as industry that the most promising results will be achieved. The expectation is that this Thesis had contributed by giving the interested reader some starting background and stimulated new research for the design of new effective approaches for unwanted traffic identification. The hope is to see other more advanced approaches for the protection of the Internet emerging in the near future.

References

- [1] CSI. Computer Security Institute. Available from <http://www.gocsi.com>.
- [2] L. A. Gordon, M. P. Loeb, W. Lucyshyn, and R. Rich. 2005 CSI/FBI Computer Crime Survey. In *10th Annual Computer Crime and Security*, 2005.
- [3] L. A. Gordon, M. P. Loeb, W. Lucyshyn, and R. Rich. 2006 CSI/FBI Computer Crime Survey. In *11th Annual Computer Crime and Security*, 2006.
- [4] R. Richardson. 2007 CSI/FBI Computer Crime Survey. In *12th Annual Computer Crime and Security*, 2007.
- [5] Gartner. Gartner. Available from <http://www.gartner.com>.
- [6] Radicati Group. Corporate Email Market 2006-2010. Available from <http://www.radicati.com>.
- [7] F. Ricciato. Unwanted traffic in 3G Network. *SIGCOMM Computer Communications Review*, 36(2):53-56, April 2006.
- [8] CERT.br. Computer Emergency Response Team Brazil. Available from <http://www.cert.br>.
- [9] CAIS. RNP's Security Incident Response Team. Available from <http://www.rnp.br/cais>.
- [10] L. Anderson, E. Davies, and L. Zhang. Report from the IAB workshop on Unwanted Traffic March 9-10 2006. IETF, Internet informational RFC 4948, 2007.
- [11] B. Krishnamurthy. Unwanted traffic: Important problems, research approaches. In *Internet Architecture Board Workshop*, 2006.
- [12] E. Davies. Unwanted Traffic. *IETF Journal*, December 2007.
- [13] E. H. Spafford. The internet worm program: an analysis. *SIGCOMM Computer Communications Review*, 19(1):17-57, January 1989.
- [14] CNN. Cyber-attacks batter Web heavyweights. Available from <http://www.cnn.com/2000/TECH/computing/02/09/cyber.attacks.01/index.html>
- [15] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson. Characteristics of internet background radiation. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, pages 27-40, 2004.
- [16] P. Soto. Identifying and Modeling Unwanted Traffic on the Internet. Massachusetts Institute of Technology, M.Sc. Thesis, 2005.
- [17] K. Xu, Z-L. Zhang, and S. Bhattacharyya. Reducing unwanted traffic in a backbone Network. In *Steps of Reducing Unwanted Traffic on the Internet Workshop (SRUTI)*, 2005.

- [18] J. Yarden. Case study: How much does unwanted Internet traffic really cost an organization? Available from http://articles.techrepublic.com.com/5100-10878_11-5967393.html.
- [19] E. L. Feitosa, E. Souto, and D. Sadok. Unwanted Internet Traffic: Concepts, Characterization, and Solutions. In *Text-Book of Mini courses of the VIII of the Brazilian Symposium on Information Security and Computing Systems (SBSeg'08)*. Porto Alegre, Brasil: SBC, 2008, ch. 3.
- [20] Skype. Available from <http://www.skype.com>.
- [21] A. Brodsky. Comcast Case Is A Victory for the Internet. Available from <http://www.publicknowledge.org/node/1686>.
- [22] F-Secure. F-Secure Malware Information Pages: Small.DAM. Available from http://www.f-secure.com/v-descs/small_dam.shtml.
- [23] H. Schulze and K. Mochalski. Ipoque Internet Study 2008/2009. Available from http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009.
- [24] Wikipedia. Samy (XSS). Available from [http://en.wikipedia.org/wiki/Samy_\(XSS\)](http://en.wikipedia.org/wiki/Samy_(XSS)).
- [25] D. Clark. The design philosophy of the DARPA internet protocols. In *Symposium Proceedings on Communications Architectures and Protocols*, pages 106-114, 1988.
- [26] B. Carpenter. Architectural principles of the Internet. IETF, Internet informational RFC1958, 1996.
- [27] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277-288, November 1984.
- [28] R. Bush and D. Meyer. Some Internet Architectural Guidelines and Philosophy. IETF, Internet informational RFC 3439, 2002.
- [29] RIPE NCC. YouTube Hijacking: A RIPE NCC RIS case study. Available from <http://www.ripe.net/news/study-youtube-hijacking.html>.
- [30] M. S. Blumenthal and D. D. Clark. Rethinking the design of the Internet: the end-to-end arguments vs. the brave new world. *ACM Transactions on Internet Technology*, 1(1):70-109, August 2001.
- [31] Kaspersky. Kaspersky Lab. Available from <http://www.kaspersky.com>.
- [32] P. Mockapetris. Domain Names - Concepts and Facilities. IETF, Internet standard RFC 1034, 1987.
- [33] P. Mockapetris. Domain Names - Implementation and Specification. IETF, Internet standard RFC 1035, 1987.
- [34] G. Lawton. Stronger Domain Name System Thwarts Root-Server Attacks. *IEEE Computer Society*, 40(5):14-17, 2007.
- [35] J. Fontana. Network World Fusion. Available from <http://www.networkworld.com/news/2001/0125mshacked.html>.
- [36] RIPE. RIPE Mail Archive. Available from <https://www.ripe.net/ripe/maillists/archives/eof-list/2002/msg00009.html>.

- [37] J. Vijayan. Akamai Attack Reveals Increased Sophistication. Available from <http://www.landfield.com/isn/mail-archive/2004/Jun/0088.html>.
- [38] B. Krebs. Blue Security Kicked While It's Down. Available from http://blog.washingtonpost.com/securityfix/2006/05/blue_security_surrenders_but_s.html.
- [39] M. Olney, P. Mullen, and K. Miklavcic. Dan Kaminsky's 2008 DNS Vulnerability. Available from http://www.snort.org/vrt/docs/white_papers/DNS_Vulnerability.pdf.
- [40] R. McMillan. DNS Trouble Knocks NSA off Internet. Available from http://www.cio.com/article/358513/DNS_Trouble_Knocks_NSA_Off_Internet.
- [41] R. Naraine. Websense reports China Netcom DNS cache poisoning. Available from <http://blogs.zdnet.com/security/?p=1776#more-1776>.
- [42] HoneyNet Project. Know Your Enemy: Fast-Flux Service Networks. Available from <http://www.honeynet.org/papers/ff/>. 2007.
- [43] F-Secure. F-Secure Malware Information Pages: WareZov. Available from <http://www.f-secure.com/v-descs/warezov.shtml>. 2006
- [44] Y. Wang, D. Beck, J. Wang, C. Verbowski, and B. Daniels. Strider Typo-Patrol: Discovery and Analysis of Systematic Typo-Squatting. In *2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI'06)*, pages 5-10, 2006
- [45] D. Kravets. Google Profits From Typo Squatting, Report Charges. Available from <http://blog.wired.com/27bstroke6/2008/10/google-profitin.html>. 2008.
- [46] C. Jackson, A. Barth, A. Bortz, W. Shao, and D. Boneh. Protecting Browsers from DNS Rebinding Attacks. *ACM Transactions on the Web (TWEB)*, 3(1):1-26, January 2009.
- [47] B. Zdrnja, N. Brownlee, and D. Wessels. Passive Monitoring of DNS Anomalies. In *4th Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, pages 129–39, 2007.
- [48] Rekhter, Y., Li, T., and Hares, S. A Border Gateway Protocol 4 (BGP-4). IETF, Internet draft standard RFC 4271, January 2006.
- [49] Arbor Networks. Worldwide ISP Security Report. Available from <http://www.arbornetworks.com>. 2005.
- [50] R. Kuhn, K. Sriram, and D. Montgomery. Border Gateway Protocol Security. Available from <http://csrc.nist.gov/publications/nistpubs/800-54/SP800-54.pdf>. 2007.
- [51] O. Nordström and C. Dovrolis. Beware of BGP attacks. *SIGCOMM Computer Communications Review*, 34(2):1-8, April 2004.
- [52] R. Farrow. Network Defense, Routing Instability, Border Gateway Protocol, the routing glue of the Internet, Lacks Strong Security. Available from <http://www.spirit.com/Network/net0102.html>. 2002.
- [53] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. F. Wu, and L. Zhang. An analysis of BGP multiple origin AS (MOAS) conflicts. In *1st ACM SIGCOMM Workshop on Internet Measurement*, pages 31-35, 2001.

- [54] M. Kassner. BGP: Yet another Internet time bomb. Available from <http://blogs.techrepublic.com.com/networking/?p=663>. 2008.
- [55] D. Kaminsky. The emergency of a theme. Available from <http://www.doxpara.com/?p=1231>. 2008.
- [56] MAAWG. Email Metrics Program: The Network Operators' Perspective. Available from http://www.maawg.org/about/FINAL_1Q2006_Metrics_Report.pdf. 2006.
- [57] ENISA. Survey on Industry Measures taken to comply with National Measures implementing Provisions of the Regulatory Framework for Electronic Communications relating to the Security of Services. Available from http://www.enisa.eu.int/doc/pdf/deliverables/enisa_security_spam.pdf/. 2006.
- [58] APWG. Phishing Activity Trends Report. Available from http://www.antiphishing.org/reports/apwg_report_feb_06.pdf. 2006.
- [59] P. D. Kretkowski. Brace Yourself: VoIP Spam Is Coming. Available from <http://www.voip-news.com/feature/voip-spam-spit-021207>. 2007.
- [60] S. Ahson and M. Ilyas. *SIP Handbook: Services, Technologies, and Security of Session Initiation Protocol*: CRC Press, 2008.
- [61] M. Desantis. Understanding Voice over Internet Protocol (VoIP). Available from http://www.us-cert.gov/reading_room/understanding_voip.pdf. 2008.
- [62] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. IETF, Internet proposed standard RFC 3261, 2002.
- [63] Arbor Networks. Atlas. Available from <http://atlas.arbor.net>. 2009.
- [64] Shadowserver Foundation. Available from <http://www.shadowserver.org>. 2009.
- [65] New York Time. Facebook, Inc. News. Available from http://topics.nytimes.com/top/news/business/companies/facebook_inc/index.html. June, 21 2010.
- [66] ENISA. Security Issues and Recommendations for Online Social Networks. Available from http://www.enisa.europa.eu/doc/pdf/deliverables/enisa_pp_social_networks.pdf. 2007.
- [67] FriendBot. MySpace friend adder. Available from <http://www.friendbot.com>. 2008.
- [68] Wikipedia. Cross site scripting. Available from http://en.wikipedia.org/wiki/Cross-site_scripting. 2008.
- [69] J. Grossman. Cross-Site scripting worms and viruses. Available from <http://www.whitehatsec.com/downloads/WHXSSThreats.pdf>. 2006.
- [70] F-Secure. F-Secure Malware Information Pages: JS/QuickSpace.A. Available from http://www.f-secure.com/v-descs/js_quickspace_a.shtml. 2007.
- [71] PACKETEER. Keeping an Eye on Recreational Traffic. Available from <http://networkthatthinks.com/downloads/PKTR-ProbTraffic-LifeCycle-vA.pdf>. 2007.

- [72] E. Hjelmvik and W. John. Breaking and Improving Protocol Obfuscation. Department of Computer Science and Engineering, Chalmers University of Technology, Technical Report No. 2010-05, ISSN 1652-926X, 2010.
- [73] R. Zhang and M. Bartell. *BGP Design and Implementation*. Cisco Press, 2004
- [74] M. Wu, R. C. Miller, and G. Little. Web Wallet: Preventing Phishing Attacks by Revealing User Intentions. In *Symposium on Usable Privacy and Security (SOUPS)*, pages 102-113, 2006.
- [75] N. Provos. Honeyd. Available from <http://www.honeyd.org>. 2009.
- [76] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which employs IP Source Address Spoofing. IETF, Internet best currency practices RFC 2827, 2000.
- [77] F. Baker and P. Savola. Ingress Filtering for Multihomed Networks. IETF, Internet best currency practices RFC 3704, 2004.
- [78] W. Kumari and D. McPherson. Remote Triggered Black Hole filtering with uRPF. IETF, Internet draft, 2009.
- [79] Cisco. Remotely Triggered Black Hole Filtering – Destination Based and Source Based. Cisco System, White Paper Available from <http://www.cisco.com/web/about/security/intelligence/blackhole.pdf>. 2005.
- [80] E. Cooke, M. Bailey, D. Watson, F. Jahanian, and J. Nazario. The Internet Motion Sensor: A Distributed Blackhole Monitoring System. University of Michigan, Technical Report CSE-TR-491-04, 2004.
- [81] M. Bailey, E. Cooke, F. Jahanian, J. Nazario, and D. Watson. The Internet Motion Sensor: A Distributed Blackhole Monitoring System. In *12th Annual Network and Distributed System Security Symposium (NDSS'05)*, pages 167-179, 2005.
- [82] D. Moore. Network Telescopes: Observing Small or Distant Security Events. In *11th USENIX Security Symposium*, 2002.
- [83] D. Moore, C. Shannon, G. M. Voelkery, and S. Savage. Network Telescopes: Technical Report. Cooperative Association for Internet Data Analysis (CAIDA), Technical Report tr-2004-04, 2004.
- [84] J. Roschelle and S. D. Teasley. The construction of shared knowledge in collaborative problem solving. In C. O'Malley, editor, *Computer Supported Collaborative Learning*, pages 67-97. Berlin: Springer. 1995.
- [85] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the slammer worm. *IEEE Security & Privacy Magazine*, 1(4): 33-39, 2003.
- [86] Symantec. Outbreak alert: storm trojan. Available from http://www.symantec.com/outbreak/storm_trojan.html. 2007.
- [87] D. McPherson. 2% of Internet traffic raw sewage. Available from <http://asert.arbornetworks.com/2008/03/2-of-internet-traffic-raw-sewage>. 2008.
- [88] H., Curry, D., Feinstein, B. Debar. The Intrusion Detection Message Exchange Format (IDMEF). IETF, Internet Experimental RFC 4765, 2007.
- [89] R. Danyliw, J. Meijer, and Y. Demchenko. The Incident Object Description Exchange Format. IETF, Internet proposed standard RFC 5070, 2007.

- [90] B. Feinstein and G. Matthews. The Intrusion Detection Exchange Protocol. IETF, Internet experimental RFC 4767, 2007.
- [91] J. Yu, Y. V. Ramana Reddy, S. Selliah, S. Kankanahalli, S. Reddy, and V. Bharadwaj. TRINETR: an intrusion detection alert management systems. In *13th IEEE International Workshops on Enabling Technologies*, pages. 235-240, 2004.
- [92] J. E. M. S. Brandão, J. S. Fraga, P. M. Mafra, and R. R. Obelheiro. A WS-Based Infrastructure for Integrating Intrusion Detection Systems in Large-Scale Environments. In *CoopIS/DOA/ODBASE/GADA*, 2006.
- [93] R. Sadoddin and A. Ghorbani. Alert correlation survey: framework and techniques. In *2006 international Conference on Privacy, Security and Trust - PST '06*, 2006.
- [94] A. Valdes and K. Skinner, Probabilistic Alert Correlation. In *Recent Advances in Intrusion Detection (RAID)*, 2001.
- [95] K. Julicsh. Mining Alarm Clusters to Improve Alarm Handling Efficiency. In *17th Annual Conference on Computer Security Applications*, 2003.
- [96] F Cuppens. Managing alerts in a multi-intrusion detection environment. In *17th Annual Conference on Computer Security Applications (ACSAC)*, pages 22-31, 2001.
- [97] B. Morin, L. Me, H. Debar, and M. Ducasse. M2D2: a formal data model for IDS alert correlation. In *Recent Advances in Intrusion Detection (RAID)*, pages 15-37, 2002.
- [98] Y. Xie, H. Kim, D. O. Hallaron, M. Reiter, and H. Zhang. Seurat: a point list approach to anomaly detection. In *7th International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2004.
- [99] C. V. Zhou, C. Leckie, and S. Karunasekera. A survey of coordinated attacks and collaborative intrusion detection. *Computer & Security*, 2009.
- [100] R. Yusof, S. R. Selamat, and S. Sahib. Intrusion Alert Correlation Technique Analysis for Heterogeneous Log. *International Journal of Computer Science and Network Security*, 8(9), September 2008.
- [101] P. A. Porras and P. G. Neumann. EMERALD: Event monitoring enabling responses to anomalous live disturbances. In *20th National Information System Security Conference*, pages 353-357, 1997.
- [102] H. Debar and A. Wespi. Aggregation and Correlation of Intrusion-Detection Alerts. In *Recent Advances in Intrusion Detection (RAID)*, 2001.
- [103] B. Zhu and A. A. Ghorbani. Alert correlation for extracting attacks strategies. *International Journal of Network Security*, 3(2):244-258, 2006.
- [104] B. Morin and H. Debar. Correlation of Intrusion Symptoms: an Application of Chronicles. In *6th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 94-112, 2003.
- [105] C Dousson. Suivi d'évolutions et reconnaissance de chroniques. PhD Thesis, 1994.

- [106] F. Cuppens and R. Ortalo. LAMBDA: A language to model a database for detection attacks. In *3th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 197-216, 2000.
- [107] S. Eckmann, G. Vigna, and R. Kemmerer. STATL: An attack language for state-based intrusion detection, 2002.
- [108] R. Vankamamidi. ASL: A specification language for intrusion detection and network monitoring. Iowa State University, M.Sc. Thesis, 1998.
- [109] S. Templeton and L. Levitt. A requires/provides model for computer attacks. In *New security paradigms workshop*, pages 31-38, 2000.
- [110] E. Totel, B. Vivinis, and L. Mé. A language driven ids for event and alert correlation. In *SEC*, pages 209-224, 2004.
- [111] O. Dain and R. Cunningham. Fusing a heterogeneous alert stream into scenarios. In *2001 ACM workshop on data mining for security applications*, pages 1-13, 2001.
- [112] P. Ning, Y. Cui, and D. Reeves. Constructing attack scenarios through correlation of intrusion alerts. In *ACM Conference of Computer and Communications Security*, pages 245-254, 2002.
- [113] X. Qui and W. Le. Statistical Causality of INFOSEC Alert Data. In *6th International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2003.
- [114] X. Qui. A Probabilistic-Based Framework for INFOSEC Alert Correlation. Georgia Institute of Technology, PhD Thesis, 2005.
- [115] M. Almgren, U. Lindqvist, and E. Jonsson. A multi-sensor model to improve automated attack detection. In *11th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 291-310, 2008.
- [116] A. Alharby and H. Imai. IDS false alarm reduction using continuous and discontinuous patterns. In *ACNS*, pages 192-205, 2005.
- [117] J. Viinikka and H. Debar. Monitoring IDS background noise using EWMA control charts and alert information, pages 166-187, 2004.
- [118] S. Manganaris, M. Christensen, D. Zerkle, and K. Hermiz. A data mining analysis of rtid alarms. *Computer Networks*, 34(4):517-524, 2000.
- [119] H. Mannila and H. Toivinen. Discovering Generalized Episodes using Minimal Occurrences. In *Second International Conference on Knowledge Discovery and Datamining*, 1996.
- [120] H. Mannila, H. Toivonen, and A. Inkeri Verkamo. Discovery of Frequent Episodes in Event Sequences. *Data Mining and Knowledge Discovery*, 1(3):259-289, January 1997.
- [121] C. Clifton and G. Gengoo. Developing custom intrusion detection filters using data mining. In *21st century military communications MILCOM*, pages 440-442, 2000.
- [122] D. Yu and D. A. Fincke. Alert correlation fusion in intrusion detection system with extended dempster-shafer theory. In *IEEE Symposium on Security and Privacy*, 2005.

- [123] H. Svensson and A. Josang. Correlation intrusion alarms with subjective logic. In *Proceedings of the 6th Nordic Workshop on Secure IT systems (NordSec2001)*, Copenhagen, Denmark, November 2001.
- [124] T. Pietraszek. Using adaptive alert classification to reduce false positives in intrusion detection. In *Symposium on Recent Advances in Intrusion Detection (RAID '04)*, pages 102–124, 2004.
- [125] T. Pietraszek and A. Tanner. Data mining and machine learning - towards reducing false positives in intrusion detection. *Information Security Tech Rep*, 10(3):169–183, 2005.
- [126] Z. Tian, W. Zhang, J. Ye, X. Yu, and H. Zhang. Reduction of false positives in intrusion detection via adaptive alert classifier. In *International Conference on Information and Automation (ICIA)*, pages 1599–1602, 2008.
- [127] R. Alshammari, S. Sonamthiang, M. Teimouri, and D. Riordan. Using neuro-fuzzy approach to reduce false positive alerts. In *5th annual conference on communication networks and services research*, pages 345-349, 2007.
- [128] E. Hooper. An intelligent detection and response strategy to false positives and network attacks. In *4th IEEE International Workshop on Information Assurance (IWIA 2006)*, pages 20–31, 2006.
- [129] P. A. Porras, M. W. Fong, and A. Valdes. A Mission-Impact-Based Approach to INFOSEC Alarm Correlation. In *5th International Symposium on Recent Advances in Intrusion Detection (RAID 2002)*, pages 95–114, 2002.
- [130] K. Alsubhi, E. Al-Shaer, and R. Boutaba. Alert Prioritization in Intrusion Detection Systems. In *IEEE/IFIP Network Operations and Management Symposium: Pervasive Management for Ubiquitous Networks and Services (NOMS 2008)*, Salvador, Brasil, pages 7-11, 2008.
- [131] N. Ye, Q. Chen, and C. M. Borrer. EWMA Forecast of Normal System Activity for Computer Intrusion Detection. *IEEE Transactions on Reliability*, 53(4):557–566, December 2004.
- [132] P. Z. Hu and M. I. Heywood. Predicting intrusions with local linear model. In *Proceedings of the International Joint Conference on Neural Networks*, 3:1780–1785, July 2003.
- [133] L. Wang, A. Liu, and S. Jajodia. Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. *Computer Communications*, 29:2917-2933, 2006.
- [134] P. Kannadiga, M. Zulkernine, and A. Haqu. E-NIPS: An Event-Based Network Intrusion Prediction System. In *ISC*, pages 37-52, 2007.
- [135] A. Scherrer, N. Larrieu, P. Owezarski, and P. Borgnat. Non-Gaussian and Long Memory Statistical Characterizations for Internet Traffic with Anomalies. *IEEE Transactions on Dependable and Secure Computing*, 4(1):56-70, 2007.
- [136] A. Scherrer, N. Larrieu, P. Borgnat, P. Owezarski, and P. Abry. Non Gaussian and Long Memory Statistical Modeling of Internet Traffic. In *4th International Workshop on Internet Performance, Simulation, Monitoring and Measurement (IPS-MoMe)*, Salzburg, Austria, 2006.

- [137] Lawrence Berkeley National Laboratory. The Internet Traffic Archive. Available from <http://ita.ee.lbl.gov>. 2009.
- [138] WAND. WAND Network Resource Group. Available from <http://wand.cs.waikato.ac.nz/wits/auck/4>. 2009.
- [139] CAIDA. Available from <http://www.caida.org/data>. 2009.
- [140] UNC/FORTH. UNC/FORTH Archive of Wireless Traces, Models, and Tools. Available from <http://netserver.ics.forth.gr/datatraces>. 2009.
- [141] GIP Renater. Available from <http://www.renater.fr>. 2009.
- [142] G. Dewaele, K. Fukuda, P. Borgnat, P. Abry, and K. Cho. Extracting hidden anomalies using sketch and non Gaussian multiresolution statistical detection procedures. In *2007 Workshop on Large Scale Attack Defense*, Kyoto, Japan, pages 145-152, 2007.
- [143] P. C. Mahalanobis. On tests and measures of groups divergence. *Journal of the Asiatic Society of Bengal*, vol. 26, 1930.
- [144] P. Abry, P. Borgnat, and G. Dewaele. Sketch based anomaly detection, identification and performance evaluation. In *IEEE/IPSJ SAINT Measurement Workshop*, 2007.
- [145] K. Cho, K. Mitsuya, and A. Kato. Traffic data repository at the WIDE project . In *Annual Conference on USENIX*, San Diego, California, 2000.
- [146] O. Salem, S. Vaton, and A. Gravey. A Novel Approach for Anomaly Detection for High-Speed Networks. In *3th European Conference on Computer Network Defense*, 2007.
- [147] R. Schweller, L. Zhichun; Y. Chen, Y. Gao, A. Gupta, Y. Zhang, P. A. K. Ming-Yang, and G. Memik. Reversible sketches: enabling monitoring and analysis over high-speed data streams. *IEEE/ACM Transactions Network*, 15(5):1059-1072, 2007.
- [148] S. Fluhrer and D. McGrew. Statistical analysis of the alleged RC4 keystream generator. In *7th International Workshop on Fast Software Encryption*, London, UK, pages 19-30, 2001.
- [149] Endace. Endace DAG 3.6ET. Available from <http://www.endace.com/dag-network-monitoring-cards.html>. 2008.
- [150] J. Gao, G. Hu, X. Yao, and R. Chang. Anomaly Detection of Network Traffic Based on Wavelet Packet. In *Asia-Pacific Conference*, 2006.
- [151] K. Xu, Z. Zhang, and S. Bhattacharyya. Profiling internet backbone traffic: behavior models and applications. In *2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '05)*, Philadelphia, Pennsylvania, USA, pages 169-180, 2005.
- [152] C. E. Shannon and T. Weaver. *The Mathematic Theory of Communication*. University of Illinois Press, 1949.
- [153] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. In *2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '05)*, Philadelphia, Pennsylvania, USA, pages 217-228, 2005.

- [154] R. Dunia and S. J. Qin. A subspace approach to multidimensional fault identification and reconstruction. In *American Institute of Chemical Engineers (AIChE) Journal*, pages 1813–1831, 1998.
- [155] A. Lakhina, M. Crovella, and C. Diot, Diagnosing network-wide traffic anomalies. In *2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'04)*, Portland, Oregon, USA, pages 219-230, 2004.
- [156] Abilene Global Research Network Operations Center. Available from <http://www.abilene.iu.edu>.
- [157] GEANT Project. Available from <http://www.geant.net/pages/home.aspx>.
- [158] R. R. Aschoff. ChkModel: Um Mecanismo de Defesa Contra Ataques DDoS. Federal University of Pernambuco, Undergraduate Thesis, 2007.
- [159] R. W. Stevens. UNIX Network Programming. Second Edition: Networking APIs: Sockets and XTI. Prentice Hall, 1998.
- [160] Force10 Networks. P-Series Overview. Available from <http://www.force10networks.com/products/pseries.asp>. 2008.
- [161] Cetacea Networks. OrcaFlow: Terabit-Class Network Traffic Anomaly Detection. Available from <http://www.orcaflow.ca/orcaflow-ca>. 2008.
- [162] CloudShield Technologies. Hardware Solutions. Available from <http://www.cloudshield.com/platform/hardware.asp>. 2008.
- [163] Snort. Available from <http://www.snort.org>. 2009.
- [164] NFS. Bro Intrusion Detection System. Available from <http://bro-ids.org>. 2009.
- [165] Prelude-IDS Technologies. Prelude-IDS. Available from <http://www.prelude-ids.com>. 2009.
- [166] Nepenthes. Available from <http://nepenthes.carnivore.it>. 2009.
- [167] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: multilevel traffic classification in the dark. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Philadelphia, Pennsylvania, USA, pages 229-240, 2005.
- [168] OASIS. UDDI Version 3.0.2. Available from <http://www.oasis-open.org/specs/index.php#uddiv3>. 2005.
- [169] L. Zhi-tang, L. Yao, and W. Li. A Novel Fuzzy Anomaly Detection Algorithm Based on Artificial Immune System. In *8th International Conference on High-Performance Computing in Asia-Pacific Region*, Washington, DC, 2005.
- [170] G. Xiang, W. Min, and Z. Rongchun. Application of Fuzzy ART for Unsupervised Anomaly Detection System. In *International Conference on Computational Intelligence and Security*, pages 621-624, 2006.
- [171] C. Siaterlis, B. Maglaris, and P. Roris. A novel approach for a Distributed Denial of Service Detection Engine. National Technical University of Athens., Technical Report, 2003.

- [172] Y. Chen, K. Hwang, and W. Ku. Collaborative Detection of DDoS Attacks over Multiple Network Domains. *IEEE Transactions on Parallel and Distributed Systems*, 18(12):1649-1662, December 2007.
- [173] D. Sadok, E. Souto, E. Feitosa, J. Kelner, L. Westberg. RIP – A robust IP access architecture. *Computers & Security*, 28(6):359-380, 2009.
- [174] A. Westerinen, J. Schnizlein, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, and S. Waldbusser. Terminology for *Policy-Based Management*. IETF, Internet informational RFC 3198, 2001.
- [175] D. C. Verma. *Policy-Based Networking: Architecture and Algorithms*. New Riders Publishing, 2000.
- [176] D. E. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13(2):222-232, February 1987.
- [177] EJ-Technologies. JProfiler. Available from <http://www.ej-technologies.com/products/jprofiler/overview.html>. 2010.
- [178] Snort IDMEF Plugin. Available from <http://sourceforge.net/projects/snort-idmef/>. 2010.
- [179] MIT Lincoln Laboratory. Darpa Intrusion Detection Scenario Specific Data Sets. Available from <http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html>. 2000.
- [180] C. V. Zhou, C. Leckie and S. Karunasekera. Decentralized multi-dimensional alert correlation for collaborative intrusion detection. In *Journal of Network and Computer Applications*, 32:1106–1123, 2009.
- [181] W. Lee, S. J. Stolfo, and K. Mok. Adaptive Intrusion Detection: A Data Mining Approach. *Artificial Intelligence Reviews*, 14(6):533-567, December 2000.
- [182] J. Luo and S. M. Bridges. Mining fuzzy association rules and fuzzy frequent episodes for intrusion detection. *International Journal of Intelligent Systems*, 15(8):687–703, 2000.
- [183] J. Luo, M. S. Bridges, and B. R. Vaughn. Fuzzy frequent episodes for real-time intrusion detection. In *Proceedings of the IEEE international conference on fuzzy systems*, 1:368–371, 2001.
- [184] M. Qin and K. Hwang. Frequent Episode Rules for Intrusive Anomaly Detection with Internet Datamining. In *USENIX Security Symposium*, 2004.
- [185] K. Hwang, M. Cai, Y. Chen, and M. Qin. Hybrid Intrusion Detection with Weighted Signature Generation over Anomalous Internet Episodes. *IEEE Transactions on Dependable and Secure Computing*, 4(1):41-55, 2007.
- [186] M. Soleimani and A. A. Ghorbani. Critical Episode Mining in Intrusion Detection Alerts. In *CNSR*, pages 157-164, 2008.
- [187] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, New York, 1999.
- [188] Y. Yao. Information Retrieval Support System. In *IEEE World Congress on Computational Intelligence*, pages 773-778, 2002.

- [189] J. Yao and Y. Yao. Web-based Information Retrieval Support System: Building Research Tools for Scientists in the New Information Age. In *IEEE/WIC International Conference on Web Intelligence*, pages 570-573, 2003.
- [190] J. Yao and Y. Yao. Information granulation for Web based information retrieval support systems. In *Proceedings of SPIE*, 5098:138-146, 2003.
- [191] O. Hoerber. Web Information Retrieval Support Systems: The Future of Web Search. In *2008 International Workshop on Web Information Retrieval Support Systems*, pages 29-32, 2008.
- [192] Google. Google Maps. Available from <http://maps.google.com>. 2009.
- [193] AllInOneNews. Available from <http://www.allinonenews.com>. 2009.
- [194] Y. Zeng, Y. Yao, and N. Zhong. DBLP-SSE: A DBLP Search Support Engine. In *IEEE/WIC/ACM International Conference on Web Intelligence*, 2009.
- [195] G. Marchionini and R. W. White. Information Seeking Support System. *IEEE Computer*, 42(3):30-32, March 2009.
- [196] M. Ley, "The DBLP computer science bibliography: Evolution, research issues, perspectives. In *9th International Symposium of String Processing and Information Retrieval*, 2002, pp. 1-10.
- [197] M. Tilsner, O. Hoerber, and A. Fiech. CubanSea: Cluster-Based Visualization of Search Results. *IEEE Computer*, 42(3):108-112, March 2009.
- [198] C. Shah. ContextMiner: Explore Globally, Aggregate Locally. *IEEE Computer*, 42(3):94, March 2009.
- [199] WolframAlpha. Available from <http://www.wolframalpha.com>. 2009.
- [200] R. Capra and G. Marchionini. Faceted Exploratory Search Using the Relation Browser. In *NSF Workshop on Information Seeking Support Systems*, pages 81-83, 2009.
- [201] R. Capra and G. Marchionini. Faceted Browsing, Dynamic Interfaces, and Exploratory Search: Experiences and Challenges. In *Workshop on Human-Computer Interaction and Information Retrieval (HCIR 07)*, pages 7-9, 2007.
- [202] P. Hayes and B. McBride. RDF semantics. Available from <http://www.w3.org/TR/rdf-mt>. 2004.
- [203] Secunia. Secunia Advisories. Available from <http://secunia.com/advisories>. 2009.
- [204] NIST. National Vulnerability Database (NVD). Available from <http://nvd.nist.gov>. 2009.
- [205] US-CERT. Available from <http://www.us-cert.gov>. 2009.
- [206] US-CERT. KB-CERT. Available from <https://www.kb.cert.org/vuls>. 2009.
- [207] OSVDB. Open Source Vulnerabilities Database. Available from <http://www.osvdb.org>. 2009.
- [208] IBM. ISS - Threat List. Available from <http://www.iss.net/threats/ThreatList.php>. 2009.
- [209] DragonSoft. DragonSoft Vulnerability DataBase. Available from <http://vdb.dragonsoft.com>. 2009.

- [210] SecurityFocus. Available from <http://www.securityfocus.com/vulnerabilities>. 2009.
- [211] CISCO. Cisco IronPort SenderBase Security Network. Available from <http://www.senderbase.org>. 2009.
- [212] ThreatExpert. ThreatExpert - Automated Threat Analysis. Available from <http://www.threatexpert.com>. 2009.
- [213] Team Cymru. Internet Security Research and Insight - Team Cymru. Available from <http://www.team-cymru.org>. 2009.
- [214] Luis. O. C. Borba. Um esquema de divulgação sobre informações de vulnerabilidades. Federal University of Pernambuco, Undergraduate Thesis, 2009.
- [215] C. Castilho and R. Baeza-Yates. WIRE: an Open Source Web Information Retrieval Environment. In *Workshop on Open Source Web Information Retrieval (OSWIR)*, pages 27-30, 2005.
- [216] Cwr.cl. Web Information Retrieval Environment - WIRE. Available from <http://www.cwr.cl/projects/WIRE>. 2009.
- [217] Internet ArchiveHeritrix. Available from <http://crawler.archive.org>. 2009.
- [218] Apache. Nutch. Available from <http://lucene.apache.org/nutch>. 2009.
- [219] Apache. Lucene Java. Available from <http://lucene.apache.org>. 2009.
- [220] S. Osinski and D. Weiss. The Carrot2 project. Available from <http://www.carrot2.org>. 2009.
- [221] Apache. Hadoop. Available from <http://hadoop.apache.org>. 2009.
- [222] Jericho HTML Parser. Available from <http://jericho.htmlparser.net/docs/index.html>. 2009.
- [223] C. W. Cleverdon. Report on testing and analysis of investigation into comparative efficiency of indexing systems. Aslib-Cranfield Research Report, Cranfield England, 1962.
- [224] B. F. O. Lins. ADS-Fusion: Fusão de dados para detecção de anomalias baseada na teoria da evidencia de Dempster-Shafer. Federal University of Pernambuco, Undergraduate Thesis, 2008.
- [225] B. Lins, E. L. Feitosa, D. Sadok. Aplicando a Teoria da Evidência na Detecção de Anomalias. In *XXVII Brazilian Symposium of Computer Networks and Distributed Systems (SBRC'09)*. Recife, Brasil: SBC, pages 583-596, May 2009.
- [226] A. P. Dempster, Upper and Lower Probabilities Induced by a Multivalued Mapping. *Annals Mathematics Statistics*. 38:325-339, 1967.
- [227] A. P. Dempster. Upper and Lower Probability Inferences Based on a Sample from a Finite Univariate Population. *Biometrika*. 54:515-528, 1967.
- [228] G. Shafer. A mathematical theory of evidence. Princeton, Princeton University Press, 1976.
- [229] T. Reineking. Java Dempster Shafer Library. Available from <http://sourceforge.net/projects/jds>. 2009.

- [230] OASIS. Web Services Business Process Execution Language Version 2.0. Available from <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>. 2007.
- [231] T. Erl. Service-Oriented Architecture: a Field Guide to Integrating XML and Web Services. Prentice Hall PTR, 2004.
- [232] Emerging Threates. Available from <http://www.emergingthreates.net>. 2010.
- [233] Intrusense Packit. Network injection and capture. Available from <http://www.intrusense.com/software/packit>. 2010.
- [234] Scapy. Available from <http://www.secdev.org/projects/scapy>. 2010.
- [235] Ha.ckers. Slowloris HTTP DoS. Available from <http://ha.ckers.org/slowloris>. 2010.
- [236] R. Hyatt. Keeping DNS trustworthy. *ISSA Journal*, pages 37-38, January 2006.
- [237] L. E. Oliveira, R. Aschoff, B. Lins, E. L. Feitosa, D. Sadok. Avaliação de Proteção contra Ataques de Negação de Serviço Distribuídos (DDoS) utilizando Lista de IPs Confiáveis,”. In 7th Brazilian Symposium of Information Security and Computer Systems (SBSeg 2007). Rio de Janeiro, Brasil: SBC, 2007.
- [238] Y. Robiah, S. Siti Rahayu, S. Shahrin, M. A. Faizal, M. Mohd Zaki, and R. Marliza. New Multi-Step Worm Attack Model. *Journal of Computing*, 2(1), January 2010.
- [239] CERT. CERT Advisory CA-2003-20 W32/Blaster worm. Available from <http://www.cert.org/advisories/CA-2003-20.html>. 2003.
- [240] TFTPpy. TFTPpy - A Pure Python TFTP Implementation. Available from <http://tftpy.sourceforge.net>. 2010.
- [241] E. L. Feitosa, L. E. Oliveira, B. Lins, A. Carvalho Jr., R. Melo, D. Sadok, and U. Carmo. Security Information Architecture for Automation and Control Networks. In 8th Brazilian Symposium of Information Security and Computer Systems, Rio Grande do Sul, Brasil: SBC, pages 17-30, 2008.
- [242] IEEE. Standards for Local and Metropolitan Area Networks: Port based Network Access Control. IEEE Standard 802.1X-2001, June 2001.
- [243] CAIDA. The CAIDA "DDoS Attack 2007" Dataset. Available from http://www.caida.org/data/passive/ddos-20070804_dataset.xml. 2010.
- [244] CAIDA. UCSD Network Telescope -- The Backscatter-2008 Dataset. Available from http://www.caida.org/data/passive/backscatter_2008_dataset.xml. 2010.
- [245] UMASS Trace Repository. UMASS Trace Repository. Available from <http://trace.cs.umass.edu>. 2010.
- [246] MAWI. MAWI Working Group Traffic Archive. Available from <http://mawi.wide.ad.jp/mawi>. 2010.
- [247] W3C. WSDL W3C Recommendation. Available from <http://www.w3.org/TR/wsd120-primer>. 2007.
- [248] W3C. OWL W3C Recommendation. Available from <http://www.w3.org/TR/owl-features>. 2004.

- [249] P. Lincoln, P. Porra, and V. Shmatikov. Privacy-preserving sharing and correlation of security alert. In *13th USENIX security symposium*, pages 239–254, 2004.
- [250] D. Xu and P. Ning. Privacy-preserving alert correlation: A concept hierarchy based approach. In *21st annual computer security applications conference (ACSAC)*, pages 489–498, 2005.
- [251] P. Gross, J. Parekh, and G. Kaiser. Secure selecticast for collaborative intrusion detection systems. In *3rd international workshop on distributed event-based systems (DEBS)*, 2004.
- [252] R. Janakiraman, M. Waldvogel, and Q. Zhang. Indra: A Peer-to-Peer Approach to Network Intrusion Detection and Prevention. In *IEEE WETICE 2003 Workshop on Enterprise Security*, Linz, Austria, pages 226-231, 2003.
- [253] V. Yegneswaran, P. Barford, and S. Jha. Global intrusion detection in the DOMINO overlay system. In *Network and Distributed Security Symposium (NDSS)*, 2004.
- [254] OASIS. Web services security: SOAP message security 1.0. Available from <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pfd>. 2009.
- [255] T. Imamura, B. Dillaway, and E. Simon. XML Encryption syntax and processing. W3C recommendation, 2002.
- [256] D. Eastlake, J. Reagle, and D. Solo. XML-Signature syntax and processing. IETF, Standards Track RFC 3275, 2002.