ABSTRACT

SECURITY INFORMATION MANAGEMENT WITH FRAME-BASED ATTACK REPRESENTATION AND FIRST-ORDER REASONING

by
Wei Yan

Internet has grown by several orders of magnitude in recent years, and this growth has escalated the importance of computer security. Intrusion Detection System (IDS) is used to protect computer networks. However, the overwhelming flow of log data generated by IDS hamper security administrators from uncovering new insights and hidden attack scenarios. Security Information Management (SIM) is a new growing area of interest for intrusion detection. The research work in this dissertation explores the semantics of attack behaviors and designs **F**rame-based **A**ttack **R**epresentation and **F**irst-order logic **A**utomatic **R**easoning (FAR-FAR) using linguistics and First-order Logic (FOL) based approaches. Techniques based on linguistics can provide efficient solutions to acquire semantic information from alert contexts, while FOL can tackle a wide variety of problems in attack scenario reasoning and querying. In FAR-FAR, the modified case grammar PCTCG is used to convert raw alerts into frame-structured alert streams and the alert semantic network 2-AASN is used to generate the attack scenarios, which can then inform the security administrator. Based on the alert contexts and attack ontology, Space Vector Model (SVM) is applied to categorize the intrusion stages. Furthermore, a robust Variant Packet Sending-interval Link Padding algorithm (VPSLP) is proposed to prevent links between the IDS sensors and the FAR-FAR agents from traffic analysis attacks. Recent measurements and studies demonstrated that real network traffic exhibits statistical self-similarity over several time scales. The bursty traffic anomaly detection method, Multi-Time scaling Detection (MTD), is proposed to statistically analyze network traffic's Histogram Feature Vector to detect traffic anomalies.

SECURITY INFORMATION MANAGEMENT WITH FRAME-BASED ATTACK REPRESENTATION AND FIRST-ORDER REASONING

by
Wei Yan

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Engineering

Department of Electrical and Computer Engineering

May 2005

APPROVAL PAGE

SECURITY INFORMATION MANAGEMENT WITH FRAME-BASED ATTACK REPRESENTATION AND FIRST-ORDER REASONING

Wei Yan

Dr. Edwin Hou, Dissertation Advisor — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Nirwan Ansari, Dissertation Co-advisor — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Roberto Rojas-cessa, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

Dr. Swades K. De, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

Dr. Daochuan Hung, Committee Member — Date
Associate Professor of Computer Science, NJIT

BIOGRAPHICAL SKETCH

Author: Wei Yan

Degree: Doctor of Philosophy

Date: May 2005

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Engineering,
  New Jersey Institute of Technology, Newark, NJ, 07102, USA
- Master of Computer Science,
  Tianjin Unviersity, Tianjin, People's Republic of China
- Bachelor of Science in Computer Science,
  Nanjing University of Posts and Telecommunications, Nanjing, People's Republic of China

Major: Computer Engineering

Presentations and Publications:

W. Yan, E. Hou, and N. Ansari, "Mining IDS Alerts for Intrusion Scenarios with Natural Language Processing and First-order Reasoning," *submitted to ACM SIGKDD Explorations.*

W. Yan, E. Hou, and N. Ansari, "Description Logics for an Autonomic IDS Event Analysis System," *Computer Communications*, accepted.

W. Yan, E. Hou, and N. Ansari, "Frame-based Attack Representation and Real-time First Order Logic Automatic Reasoning," *Proceedings of 3rd International Conference on Information Technology: Research and Education*, June 27-29, 2005.

W. Yan, E. Hou, and N. Ansari, "Extracting and Querying Network Attack Scenarios Knowledge in IDS Using PCTCG and Alert Semantic Networks," *Proceedings of IEEE International Conference on Communications*, Seoul, Korea, May 16-20, 2005.

W. Yan, E. Hou, and N. Ansari, "A Description Logic Based Approach for IDS Security Information Management," *Proceedings of IEEE Sarnoff Symposium*, Princeton, Paper no. 1568948862, 4 pages, April 18-19, 2005.

W. Yan, E. Hou, and N. Ansari, "Extracting Attack Strategies Using PCTCG and Semantic Networks," *Proceedings of 29th Annual IEEE Conference on Local Computer Networks*, pp. 110-117, Tampa, November 16-18, 2004.

W. Yan, "Semantic Scheme to Extract Attack Strategies for Web Service Network Security," Presented at *2004 IEEE International Workshop on IP Operations and Management*, pp. 32-40, Beijing, China, October 11-13, 2004.

W. Yan, "Semantic Approach for Attack Knowledge Extraction in Intrusion Detection Systems," Presented at *4th New York Metro Area Networking Workshop*, Paper no. 13, 5 pages, New York, September 10, 2004.

W. Yan, E. Hou, and N. Ansari, "Defending Against Traffic Analysis Attack with Link Padding for Bursty Traffic," *Proceedings of 5th International Information Assurance Workshop*, pp. 46-51, West Point, June 10-11, 2004.

E. Hou, W. Yan, and N. Ansari, "Traffic Anomaly Detection and Traffic Shaping for Self-similar Aggregated Traffic," *Proceedings of 37th Annual Conference on Information Science and Systems*, pp. 213-258, March, 2003.

ACKNOWLEDGMENT

I would like to express my sincere thanks to my advisor, Dr. Edwin Hou, and co-advisor, Dr. Nirwan Ansari, for proofreading and making corrections. This dissertation could not have been completed without their kindly guidance and support. I also like to thank all my other doctoral dissertation committee members: Dr. Daochuan Hung, Dr. Roberto Rojas-cessa, and Dr. Swades K. De, for providing me comments and suggestions that greatly enhance the quality of this thesis. I am also grateful to the Department of Electrical and Computer Engineering for providing me the financial aid necessary to complete my doctoral study at New Jersey Institute of Technology.

Special thanks also go to my parents who brought me up and encouraged me to achieve my goals. I also want to thank my wife, Meizi Li and my new born son, Eric, who are the joy of my life.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

The explosive proliferation of the Internet has extended communications to the entire world. Moreover, the past few years have seen significant growth of cyber attacks on the Internet, and network security has become a critical issue that cannot be neglected in the development of computer networks. A significant increase in the spread of viruses, worms and Trojan horses over the Internet has been observed. Acts of cyber crime such as Distributed Denial of Service (DDoS) attacks are one of the more serious problems which need to be tackled. In practice, it is nearly impossible to prevent all network attack incidents. Detection of network attacks then becomes an important reactive security function. Intrusion Detection System (IDS) has become an important tool to secure networks by detecting, alerting and responding to malicious activities. Intrusion [3] is defined as an attack in which a vulnerability is exploited or a breach (resulting in a violation of the explicit or implicit security policy of the system) is violated. Intrusion detection is the process of identifying and responding to malicious activities targeting at computing and network resources.

IDS can monitor and possibly defend against the attempts to intrude into or otherwise compromise the network systems. An ideal IDS can detect all intrusions and report only real intrusions (no false alarms). It can also provide instant overview of the type, location, and severity of the attacks. Current IDS is far from ideal, as it incur high false alarm rate.

1.1 Background in Intrusion Detection System

IDS has recently gained considerable amount of interest in the computer security community. IDS may be software-based or contains specialized hardware. The major objective is to detect attack activities that may compromise system security. Figure 1.1 shows the IDS infrastructure described in [?]. IDS inspects the contents of the network traffic to look for possible attack signatures (patterns of malicious actions). For example, a SYN flood will be noticed by an IDS when a particular host is sending SYN packets without ever attempting to complete the connection. IDS can identify this and issue the corresponding alert to notify

the security administrator. Afterwards, the defense response may be undertaken either by the administrators or the IDS itself.

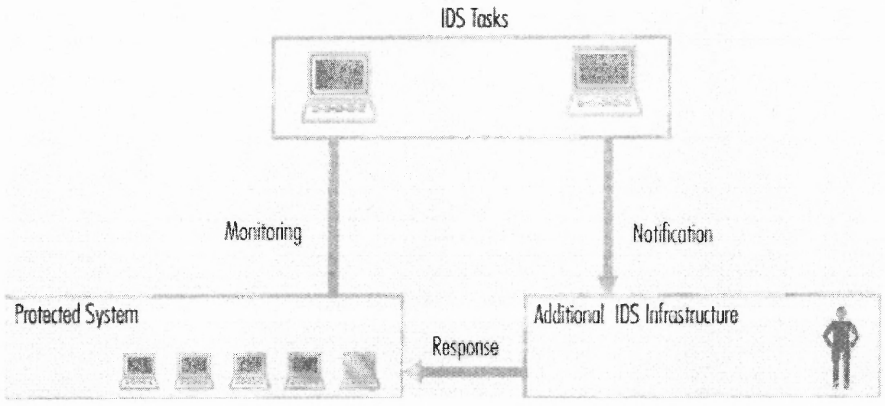


Figure 1.1 Intrusion detection system infrastructure [1].

As shown in Figure 1.2, IDS is composed of the following core components: the sensor, the attack knowledge database, and the attack response module. The sensor is an analysis engine that contains the decision-making mechanisms regarding intrusions. The detection algorithm is implemented in the sensor, which uses a script to match the text string signatures that are unique to the different intrusions. The signatures can be from a single event or a sequence of events. Another method is to distinguish the normal and abnormal behaviors, which can be done by statistical approaches. The IDS knowledge database includes the attack signatures or normal behavior profiles, and necessary parameters. The attack response module is responsible for managing the engine and presenting the attack reports.

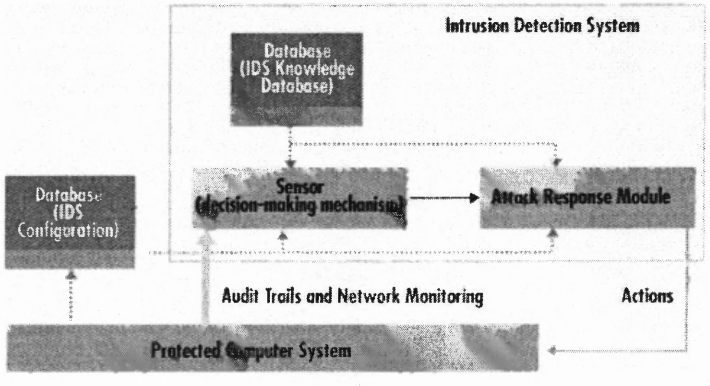


Figure 1.2 Intrusion detection system components [1].

IDS varies according to a number of criteria. IDS can be divided into Network-based IDS (NIDS)[4, 5] and Host-based IDS (HIDS) [5] on the basis of the type of systems they monitor. IDS that monitor the network for malicious traffic is called NIDS, whereas those that monitor the activities on a single host are called HIDS. NIDS consists of several sensors deployed throughout a network that report to a central console. It can monitor the network backbones and is effective at detecting outsiders attempting to penetrate the network defense. HIDS operates on hosts and can monitor the operating and file systems for signs of intrusions. Since HIDS has a limited view of only one host, it must be installed on every system being monitored.

The techniques of intrusion detection can be divided into two main detection methods: the misuse detection (such as Snort [6], Emerald [7]), and the anomaly detection (such as NADIR [8] and NIDES [9]). A misuse detection system searches for a set of known attacks that have been stored in a system database. The knowledge of the attacks is encoded as a set of signatures, which are patterns produced by corresponding attacks. The main disadvantage of misuse detection is that it can only detect the previously known intrusions, and it cannot detect new attacks. Anomaly detection detects the intrusions as a pattern recognition problem, rather than a signature matching problem. Anomaly detection builds up a set of normal behavior profiles of network traffic or hosts. Anomaly detection IDS constantly compares the newly generated profiles with the normal profiles. If it detects what it considers to be a large deviation from normal behavior, it signals an alarm to the system security administrator. With the anomaly detection, IDS can detect new attacks. The amount of false positives, however, tends to be very high with this mechanism. Contemporary commercial IDSs, such as ISS RealSecure [5], Cisco Secure IDS [10], Symantec Manhunt [11], Enterasys Dragon [4], use mostly the misuse detection methods with a model function of anomaly detection components.

Alerts generated by an IDS consist of two states: positive (indicating an intrusion) or false (not indicating an intrusion). Thus, alerts fall into four possible classifications: true positive means IDS appropriately indicates an intrusion; true negative means IDS appropriately does not indicate an intrusion; false positive means IDS inappropriately indicates an intrusion; false negative means IDS does not inappropriately indicate an intrusion. IDS should maximize

the true states, and at the same time minimize the false ones [12]. The most serious problems of IDS is the high false alarm rate (the high number of false positives), because a large number of false positives requires large amount of time to investigate.

1.2 Security Information Management

IDS generates a huge volume of alert events which are caused by elementary alerts and false alarm alerts [13, 14]. It is very time consuming and resource intensive for the security administrator to analyze this large of data and to review all the alerts, thereby hampering the performance of attack detection. Furthermore, many enterprises simply cannot afford to recruit security administrators merely for monitoring the alerts.

Within the past few years, a new security system, Security Information Management (SIM), has emerged as a solution in solving the above mentioned problems. SIM is defined as a set of processes undertaken to ensure that IDS events are collected, analyzed, and responded in the shortest period of time possible, thus ultimately reducing the administrator's reviewing time [15]. Why collect the alerts from distributed IDS sensors? With the detailed alerts stored in the multiple sensors, valuable attack information can be easily lost. By centralizing this information, events from distributed IDS sensors can be correlated and categorized.

SIM should enable a security administrator to quickly and easily find the hidden attack scenarios from the collected events, but this requires that the events from the various sources are normalized. Without the alert normalization, the processing speed at which the events can be analyzed is significantly decreased. Alert normalization requires a set of schema with a fixed number of fields to standardize incoming data regardless of its source.

Figure 1.3 shows the architecture of IDS with SIM. The logs from the firewalls, servers, and network devices are collected by SIM, where the security events are analyzed and correlated. Attack scenarios are then generated and delivered to the security administrator. SIM is composed of four parts: aggregation, normalization, correlation, and visualization. Aggregation means the reduction of redundant data. In the normalization step, SIM collects the event data from multiple IDS sensors across the network. Since the data from each security device may not be in a common format, SIM converts them into a common format. During the correlation process, SIM measures the relationship of security events to determine

the incident potential. The visualization model interface provides the capabilities of intrusion reporting for quick responses.

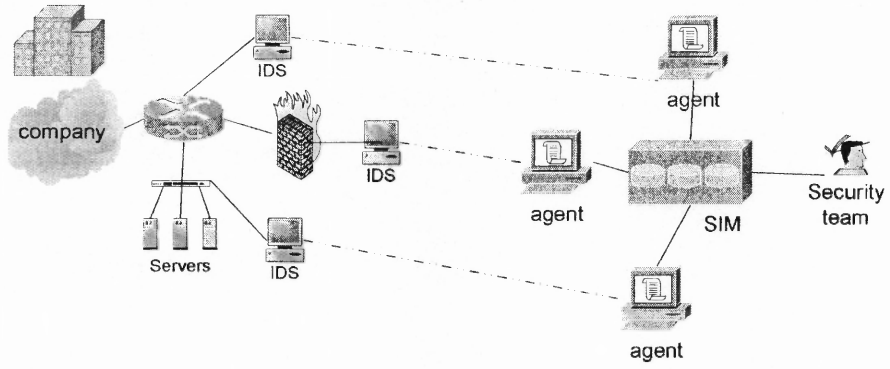


Figure 1.3 Architecture of IDS system with SIM.

Various issues have to be considered when designing SIM. First, heterogeneous IDS sensors may be installed throughout the networks. When SIM collects the event alerts from these decentralized sensors, the major obstacle is the lack of a universal alert description standard. Almost every IDS product has its own alert format, and these non-uniform alert reporting formats make alerts correlation time-consuming and difficult. Intrusion Detection Message Exchange Format (IDMEF) was proposed to be a standard IDS alert format [16]. It includes object-oriented class and a list of attributes that can describe a specific alert. For example, IDMEF includes analyzer, create time, node, user, process, and service attributes. However, IDMEF limits the alert semantic representation and reasoning because it does not provide the fields for intrusion behavior semantics, making automatic reasoning for intrusion scenarios difficult to implement. In this dissertation, IDMEF is extended to include semantic reasoning and query in capabilities.

Second, IDS can generate a large volume of alert streams, which may have a relatively high chance of being false alarms. To overcome this problem, different correlation methods [17, 18, 19, 20] were proposed to combine and analyze attack events from IDS. According to [3], intrusion correlation refers to the combination and analysis of information from all available sources about target system activities for the purposes of intrusion detection and response. However, these syntactic approaches are vendor-specific, and the heterogeneous IDS

sensors require the installations of multi-vendor correlation tools, resulting in costly network management.

Third, generally there are two methods to analyze the events: real-time analysis and off-line analysis. The goal of real-time event analysis aims to shorten the time of detecting and responding to the intrusions, thus reducing the exposure time to any given threat. In [21], the Exposure Time (ET) is equal to the Detection Time (DT) plus the Reaction Time (RT). The smaller ET is, the smaller the risk.

$$Et = Dt + Rt \tag{1.1}$$

Real-time analysis is the ideal way of protecting information assets. However, the more complex the real-time event analysis is, the more difficult it is to keep up with all the events being collected. Off-line analysis collects and analyzes data at longer time intervals, and provides more detailed results than real-time analysis. Off-line analysis can present charts and graphs to the administrators so it is easier to make the intrusion conditions more understandable. Therefore, implementing SIM should keep a balance between the benefits of real-time analysis versus the necessity for off-line analysis. In this dissertation, First-order Reasoning (FOR) is used in real-time analysis to extract the attack scenarios, whereas attack semantic query is used during the off-line analysis.

Fourth, inadequate attention has been paid to the SIM attack knowledge query interface. A well-designed interface can facilitate attack monitoring by making specific queries for the attack scenario knowledge. The traditional keyword search, such as Analysis Console for Intrusion Database [22], which asks the user to enter the keywords (e.g., IP address or alert message name, etc.) and then returns the alerts containing these items, is not suitable for the intrusion search because the number of the keyword occurrences cannot tell how relevant the search result is to the whole attack plan. For example, due to the existence of false alarms and alerts triggered by the normal network activities, one occurrence of "buffer overflow" alert may be more significant than tens of "Telnet" alerts. IDS should support attack semantic query, such as queries like "by which means the attacker controls the host or subnet, and what is the attack path from controlling the subnet to launching DDoS attacks to the application

server?" To implement such queries, the alerts' semantic information should be extracted and the alerts need to be categorized into several intrusion stages.

In this dissertation, the semantics of network intrusion behaviors are exploited and a security information management system, **F**rame-based **A**ttack **R**epresentation and **F**irst-order logic **A**utomatic **R**easoning (FAR-FAR) is proposed to reason the attack scenarios and categorize the alerts into different intrusion stages based on Natural Language Processing (NLP) and FOR. The Principal-subordinate Consequence Tagging Case Grammar (PCTCG) is used to formalize IDS alerts into frame-structured streams. The FOR tree is then used to extract the attack scenarios. Afterwards, mutual information is applied to determine the alert context range. Based on the alert contexts, the alerts are represented as the semantic space vectors. Finally, Vector Space Model (VSM) is used to categorize these vectors into different intrusion stages.

1.3 Approach Taken in This Dissertation

This dissertation describes the studies on

- exploiting NLP and FOR techniques on FAR-FAR to implement attack knowledge extraction semantic scheme;
- proposing an IDS semantic event analysis model formalized by Description Logics, which allows inferring the attack scenarios and enabling the attack knowledge semantic queries;
- using text mining techniques and vector space model to categorize the alerts into different intrusion stages;
- proposing Multi-Time scaling Detection (MTD) based on Histogram Feature Vector (HFV) to detect the bursty traffic anomalies;
- proposing Variant Packet Sending-interval Link Padding Method (VPSLP) to defend against the traffic analysis attacks based on heavy-tail distribution.

More specifically, the work in this dissertation differs from other research work in the following aspects:

- By converting the attack facts and alert streams into Conjunctive Normal Form clauses, FAR-FAR can implement both the alert correlation methods and the high-level reasoning.
- The research work in this dissertation applies the NLP method to extract the semantic concept structures from the semi-structured log alerts and represent them in a uniform way.

- The FAR-FAR semantic scheme converts alerts from "raw syntax" to "well semantics" by Description Logics
- With the attack ontology, FAR-FAR generates the semantic space vectors based on which the space vector model is applied to categorize the intrusion stages.

1.4 Basic Concepts

In this section, the terminologies used throughout this dissertation are introduced, and the definitions of the attack ontology concepts are adopted from [23, 24].

Access - establish logical or physical communication or contact.

Account - a domain of user access on a computer or network which is controlled according to a record of information which contains the user.

Action - a step taken by a user or process in order to achieve a result.

Attack - an attempt to compromise the confidentiality, integrity, availability of (the use of) information residing on a computer.

Attacker - an individual who attempts one or more attacks in order to achieve an objective.

Attack class missing focus alert number - the number of focus alerts that the attack scenario class does not include.

Attack class false focus alert number - the number of focus alerts that exist in the attack scenario class, that are not actual focus alerts.

Attack class node instance rate - the percentage of attack instance nodes in an attack scenario class.

$$Attack\ Class\ Node\ Instance\ Rate = \frac{\#\ of\ nodes\ in\ attack\ scenario\ instance}{\#\ of\ nodes\ in\ attack\ scenario\ class}$$

Attack class instance rate - the percentage of attack instance links in an attack scenario class.

$$Attack\ Class\ Link\ Instance\ Rate = \frac{\#\ of\ links\ in\ attack\ scenario\ instance}{\#\ of\ links\ in\ attack\ scenario\ class}$$

Attack class missing attack link number - the number of focus links the attack scenario class does not include.

Attack class false attack link number - the number of focus links that exist in the attack scenario class, but are not focus links.

Attack scenario - $S = \{(e_1, a_1), (e_2, a_2), \cdots (e_n, a_n)\}$ is an attack sequence of events and actions, where 2-tuple $(e_i, a_i), 1 \leq i \leq n$, which implies that attack action a_i is the primary action performed in the attack event e_i, and the effect of the event is caused by a_i.

Attack scenario class - given a sequence of the attack actions the attack scenario class is defined as all possible combinations of correlated actions with the relation weights above the semantic weight threshold. The attack scenario class includes all possible attack strategies the attacker may take.

Attack scenario instance - the subset of the attack scenario classes, generated based on the alert context.

Attack scenario missing attack links Attack scenario missing attack links means the number of focus links the attack scenario scenario does not include.

Boundary condition - a process attempts to read or write beyond a valid address boundary or a system resource is exhausted.

Buffer overflow - the classic buffer overflow results from an overflow of a static data structure.

Command - a means of exploiting a vulnerability by entering commands to a process through direct user input at the process interface.

Configuration vulnerability a vulnerability resulting from an error in the configuration of a system.

Data - facts, concepts, or instructions in a manner suitable for communication, interpretation, or processing by humans or by automatic means.

Delete - remove a target, or render it irretrievable.

Denial of Service - intentional degradation or blocking of computer or network resources.

Design vulnerability - a vulnerability inherent in the design or specification of hardware or software whereby even a perfect implementation will result in a vulnerability.

Exploits - vulnerabilities such as race conditions or undefined states in a hardware or software component that lead to performance degradation and/or system compromise.

False negative alarm - occurs when there is an attack and IDS does not raise an alarm.

False positive alarm - occurs when there is no attack and IDS raises an alarm. This case can be problematic because administrators, facing a false positive, might take unnecessary actions.

Frame - a labeled, typically with the name of the object it represents, and is made up of a series of slots. Each slot cab contains a variable number of entities.

Histogram Feature Vector - given a trace including some bins, a Histogram Feature Vector is a vector composed of the histogram frequency of every bin.

Horn clause - a statement containing at most one negative literal.

Increased access - an unauthorized increase in the domain of access on a computer or network.

Input validation - an input validation vulnerability exists if some malformed input is received by a hardware or software component and is not properly bounded or checked.

Kernel space - a process executing as part of the operating system, either compiled into the kernel or a module that is loaded into and executed by the kernel.

Local - the attacker needs to be "virtually" present at the target.

Malformed input - a process accepts syntactically incorrect input, extraneous input fields, or the process lacks the ability to handle field-value correlation errors.

Noun sequences - the noun compound or complex nominal.

Probe - access a target in order to determine its characteristics.

Process - a program in execution, consisting of the executable program, the programs data and stack, its program counter, stack pointer and other registers, and all other information needed to execute the program.

Race Condition - an error occurring during a timing window between two operations.

Remote - the attacker does not need to be "virtually" present at the target.

Root access - the attack results in the attacker having complete control of the system.

Scan - access a set of targets sequentially in order to identify which targets have a specific characteristic.

Script or program - a means of exploiting a vulnerability by entering commands to a process through the execution of a file of commands or a program at the process interface.

Spoof - masquerade by assuming the appearance of a different entity in network communications.

Toolkit - a software package which contains scripts, programs, or autonomous agents that exploit vulnerabilities.

Vulnerability - a weakness in a system allowing unauthorized action.

1.5 Dissertation Outline

The rest of the this dissertation is organized as follows:

Chapter 2 describes the related works in NLP, Semantic Web, and First-order Logics. The semantic scheme of FAR-FAR is also presented.

Chapter 3 introduces the basics of DL, presents Principal-subordinate Consequence Tagging Case Grammar, and proposes the VPSLP scheme to defend traffic analysis attacks on anonymous links between IDS sensors and agents.

Chapter 4 describes the 2-Atom Alert Semantic Network, resolution tree, alert window size and attack scenarios. It also expounds how to generate the feature vectors from alerts, and apply the vector space model to categorize the intrusion stages.

Chapter 5 describes the attack scenarios semantic query model and simulation results.

Chapter 6 statistically present the analysis of histograms of ordinary-behaving bursty traffic traces, and apply MTD to detect bursty traffic anomalies.

Chapter 7 describes the conclusions and future work.

CHAPTER 2

SEMANTICS INSPIRED SECURITY INFORMATION MANAGEMENT SYSTEM: FAR-FAR

2.1 Related Work

2.1.1 Alert Correlation

The overwhelming flow of alerts generated by IDS make it difficult for the security administrator to uncover the hidden attack scenarios. Some research studies [17, 18, 19, 20, 25] have shown that the alert correlation is an efficient solution. In [3], alert correlation is defined as the interpretation, combination, and analysis of information from all available sources about target system activities for the purposes of intrusion detection and response. The alert correlation methods can efficiently relieve the burden on the security administrator in reviewing the large number of alerts.

In [17], the aggregation and correlation component (ACC) was introduced and its purpose is to group the alerts into the duplication relationship and consequence relationship. Duplicate relationship is defined in the duplicate definition file. For example, to be considered a duplicate, the new alert's attributes must be equal to the previous alerts. The consequence relationships are defined in the consequence definition file according to the causal relations. M2D2 [25] includes four information types in the alert correlation process: the monitored system, the known vulnerabilities, the security tools, and the alerts. A mapping function is used to convert the non-formal vulnerability names into the formal ones. Furthermore, the prerequisites are modeled as remote, remote user, and local. The consequences of the vulnerability are grouped as CodeExec, DoS, and Info. This kind of classification can be viewed as a preliminary attack ontology. As a result, M2D2 aggregates alerts as "caused by the same event" and "referring to the same vulnerability". SRI [19] introduced a probabilistic approach that can handle the heterogeneous alerts based on an alert template (not IDMEF). The correlation approach considers the alert feature similarity. In [18], a new incoming alert is compared to the latest alerts in all existing scenarios, and then joins the scenario with the highest probability score. However, this method requires the attack scenarios to

be generated in advance manually and that may not be adapted to the diverse attacks and attack strategies. In [20], the correlation system uses the hyper alert that includes facts, prerequisites, and consequences of the intrusion. The hyper alerts can be correlated if the consequence of a hyper alert fulfills the prerequisites of the second hyper alert. This alert correlation method is largely based on the causal relation. However, sometimes launching certain attacks requires several steps which cannot be correlated with the causal relation.

2.1.2 Natural Language Processing and Semantic Web

There are immediate needs for improvement in information security technology that can protect networks against attacks while eliminating the needs of constant human intervention. This leads us to explore various automation techniques that can lessen the human workload. The use of state-of-the-art NLP techniques in information security has been addressed in several other researches. In [26], Raskin introduced ways in which methods and resources of NLP can be fruitfully employed in the domain of information assurance and security (IAS). He expounded that there was more incentive to apply NLP in contemporary IAS applications, such as the intelligent searching or the attack query answering. Ontological semantics [27] are employed to standardize terminology in the domain of IAS by translating non-standard terms in the texts into standard equivalents. Both the approaches and the ontology are borrowed from the field of NLP and adjusted to the needs of the security domain. Natural language information security also involves the use of NLP for intrusion prevention and detection. For example, LAMBDA [28], an attack description language is defined in a syntactic framework. The CERIAS group [29] developed some IAS applications using NLP. Furthermore, they also recommended the inclusion of NLP in IDS for more effective use of correlation engines. With the NLP model, IDS sensors can transform the input alerts into a well-defined format which can thereby be categorized.

On the other hand, Internet is primarily composed of information designed only for human to understand, but not for machine interpretation. For example, Extensible Markup Language (XML) standards provide a syntactic structure for describing data. Unfortunately, many different ways can be used to describe the same data by XML, and machines cannot unambiguously determine the correct meaning of the XML documents. In [30], Berners-Lee

described his vision of the Semantic Web, the aim of which is to add the markups to the web information and allow them readable by machines. Some previous works on semantic web have been imported in the area of semantic query. In [31], an approach for information retrieval over the Semantic Web is presented and used as the knowledge representation language. Passenger Identification, Screening, and Threat Analysis application (PISTA) [32] is developed to discover the threats for aviation safety. In [33], the authors proposed a framework for intrusion detection that is based on runtime monitoring of temporal logic specifications. The intrusion patterns are specified as formulas in First-order Logics (FOL).

2.2 FAR-FAR Semantic Scheme

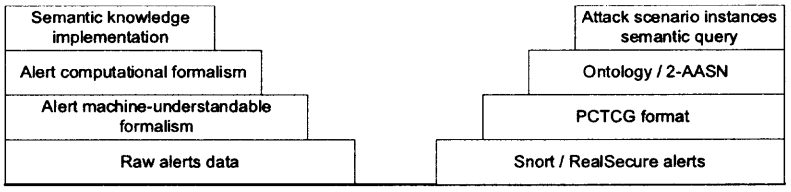


Figure 2.1 Attack knowledge representation formalism.

In this section, the attention is focused on how to combine the ideas mentioned above. Fig 2.1 represents the layered attack knowledge representation formalism, whose aim is to formalize the raw alerts into the machine-understandable, computable and finally implementable formalism. In the alert understandable layer, the syntactic-format alerts are converted into machine-understandable semantic alert streams by PCTCG and the ontology defined in the intrusion security domain. Every entity in the ontology has a corresponding element in the Description Logics (DL) formalism. Afterwards, 2-Atom Alert Semantic Network (2-AASN) was generated from PCTCG streams, and semantic operations are used over 2-AASN to generate the hidden attack scenarios. In the attack scenarios, entities and relations referenced in the ontology are translated into individuals within the DL system. Then, the spreading activation technique is used to implement the semantic attack knowledge search. Specifically, the conjunctive query is translated into a sequence of query terms using DL, and the answer to the conjunctive attack knowledge query is constituted by the set of instances of each query term.

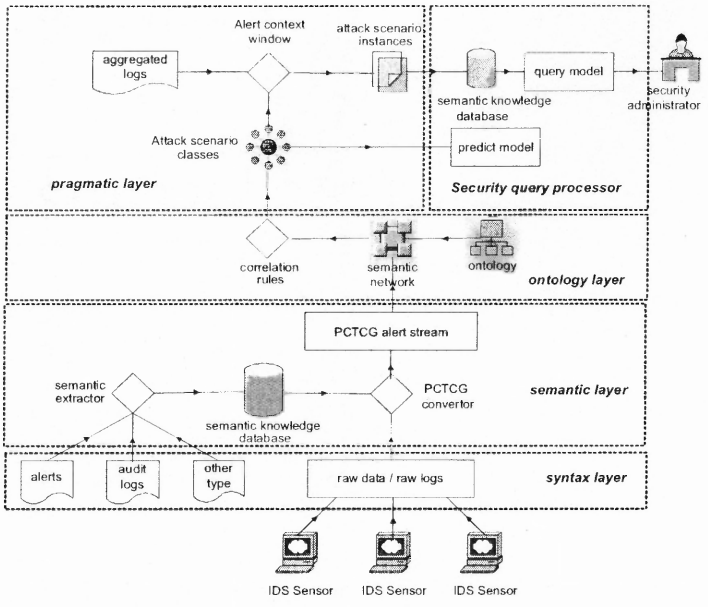


Figure 2.2 Attack knowledge extraction semantic scheme.

Figure 2.2 shows the functional architecture of the semantic scheme. It includes four layers: the syntax layer, the semantic layer, the ontology layer, and the pragmatic layer. In the syntax layer, the raw data in the form of alert log files forms the basis. We can also extend the data model to support other types of logs to make the syntax layer more compatible. IDMEF (Intrusion Detection Message Exchange Format) was proposed to be a standard alert format in the intrusion detection systems [16]. However, one principle problem of IDMEF is its use of XML (syntax structure), which limits the semantic representation. Therefore, a uniform semantic representation for the raw alerts has to be defined. The semantic layer, the ontology layer, and the semantic application layer are above the syntax layer, enabling the alerts to be understandable by the machines. In the semantic layer, there exist several semantic knowledge databases and each type of sensors has its own semantic knowledge database, where the semantic information of all the alert messages of the sensor is stored. The semantic knowledge databases are built up and maintained by the semantic extractor, which extract the semantic information from various types of raw data and stored them in the databases. The alert file and the sensor type are the inputs to the PCTCG converter. Using the semantic information in the corresponding semantic knowledge database, the PCTCG converter transfers the raw alerts into uniform PCTCG streams. Note that not all

the incoming alerts but rather the different alerts in the alert log are converted into PCTCG streams, resulting in less computation. In the ontology layer, the action-based semantic ontology is applied to the PCTCG streams to generate 2-AASN semantic networks. In the pragmatic layer (pragmatic means how semantics is used in network security monitoring), the correlation rules were applied to 2-AASN to derive the attack scenario classes for further modifications by the security administrator. In addition, the feature index table is also built up from the frame streams, that records the feature frequencies in the segmented time slots. Based on the alert contexts, the alerts are transformed into the attack space vectors and the vectors of the intrusion stages are compared to determine which intrusion the alerts belong to. Finally, the highly interpretable reasoning results can be forwarded to the security administrator.

CHAPTER 3

DESCRIPTION LOGICS, PRINCIPLE-SUBORDINATE CONSEQUENCE TAGGING CASE GRAMMAR, AND LINK PADDING

As mentioned in Chapter 2, FAR-FAR should collect the raw alerts from IDS sensors and convert the alert contents into formal representations. In FAR-FAR, the alerts are converted into semantic frame slots by PCTCG first and then the slots are represented with a formal description. In this dissertation, DL is used as the formalism for representing attack knowledge. DL is a formal logic language for representing knowledge and is the core of the knowledge representation system. Furthermore, to defend against the traffic analysis attacks on the links between the IDS sensors and SIM, the link padding approach based on heavy-tail distribution is introduced. After the link padding, the first-order statistical features of the traffic of the links are similar.

3.1 Description Logics

A knowledge representation is a fragmentary theory of intelligent reasoning. To support reasoning about the intrusion attack, a knowledge representation must describe their behavior and interactions. DL [34] is a formal language for representing knowledge and is the core of the knowledge representation system. DL systems provide their users with various inference capabilities that allow them to deduce implicit knowledge from the explicitly represented knowledge. In this section, some basic definitions of DL are introduced and the attack ontology is based on DL.

Definition 3.1.1 (Knowledge Base) *A Knowledge Base $\mathcal{KB}$ based on DL includes a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$, and is denoted as $\mathcal{KB} =< \mathcal{T}, \mathcal{A} >$. $\mathcal{T}$ contains intensional knowledge in the form of a terminology while $\mathcal{A}$ contains assertional knowledge that is specific to the individuals of the domain.*

Definition 3.1.2 (DL Interpretation) *A DL interpretation $\mathcal{I}$ is a pair $\triangle^{\mathcal{I}}$ and $\cdot^{\mathcal{I}}$, where the $\triangle^{\mathcal{I}}$ is a non-empty set called the domain of the interpretation, and $\cdot^{\mathcal{I}}$ is an interpretation function. Interpretation function $\triangle^{\mathcal{I}}$ maps*

- *each concept name A to a subset $\mathcal{A}^{\mathcal{I}}$ of $\triangle^{\mathcal{I}}$*
- *each role name R to a subset $\mathcal{R}^{\mathcal{I}}$ of $\triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$*
- *each individual name i to an element $i^{\mathcal{I}}$ of $\triangle^{\mathcal{I}}$*

Definition 3.1.3 (TBox terminological axioms) *The terminological axioms in $\mathcal{T}$ make statements about how concepts or roles are related to each other, and describe the structure of a domain. The terminological axioms have the form: $A \doteq B, A \sqsubseteq B$, and $A \bigcap B \equiv \emptyset$, where the axiom of the first kind is called equation, while the axiom of the second kind is called inclusion, and the axiom of the third kind is called disjointness. If the interpretation $\mathcal{I}$ satisfies an axiom α, let denote this as $\mathcal{I} \models \alpha$.*

- $\mathcal{I} \models A \doteq B$, *iff* $A^{\mathcal{I}} \equiv B^{\mathcal{I}}$
- $\mathcal{I} \models A \cap B \equiv \emptyset$, *iff* $A^{\mathcal{I}} \cap B^{\mathcal{I}} \equiv \emptyset$
- $\mathcal{I} \models \mathcal{T}$, *iff* $\mathcal{I}$ *satisfies every axiom in* $\mathcal{T}$.

Definition 3.1.4 (ABox assertional Axioms) *The assertional axioms "included in" have the form: $a : C$ or $< a, b >: R$, where the axiom of the first kind is called the concept assertion, while the axiom of the second kind is called the role assertion.*

- $\mathcal{I} \models a : C$, *iff* $\alpha^{\mathcal{I}} \in \mathcal{C}^{\mathcal{I}}$
- $\mathcal{I} \models < a, b >: R$, *iff* $< \alpha^{\mathcal{I}}, b^{\mathcal{I}} > \in R^{\mathcal{I}}$
- $\mathcal{I} \models A$, *iff* $\mathcal{I}$ *satisfies every axiom in* $\mathcal{I}$
- $\mathcal{I} \models \mathcal{KB} =< \mathcal{T}, \mathcal{A} >$, *iff* $\mathcal{I} \in \mathcal{A}$ *and* $\mathcal{I} \in \mathcal{T}$

Definition 3.1.5 (DL syntax and semantic) *Let A and B be concepts and R be a role. The following DL constructors build complex concepts and roles from simpler ones. The interpretation $\mathcal{I}$ can be extended to concept descriptions by the following inductive definitions.*

constructor	*syntax*	*semantics*
atomic negation	$\neg A$	$(\neg A)^{\mathcal{I}} = \triangle^{\mathcal{I}} \backslash A^{\mathcal{I}}$
intersection	$A \cap B$	$(A \cap B)^{\mathcal{I}} = A^{\mathcal{I}} \cap B^{\mathcal{I}}$
union	$A \cup B$	$(A \cup B)^{\mathcal{I}} = A^{\mathcal{I}} \cup B^{\mathcal{I}}$
value restriction	$\forall R.C$	$(\forall R.C)^{\mathcal{I}} = \{A \in \triangle^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$
exist restriction	$\exists R.C$	$(\forall R.C)^{\mathcal{I}} = \{A \in \triangle^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$
min cardinality	$\geq nR.C$	$\geq nR.C^{\mathcal{I}} = \{A \in \triangle^{\mathcal{I}} \mid \{b \mid (a,b) \in R^{\mathcal{I}}\} \mid \geq n\}$
ma cardinality	$\leq nR.C$	$\leq nR.C^{\mathcal{I}} = \{A \in \triangle^{\mathcal{I}} \mid \{b \mid (a,b) \in R^{\mathcal{I}}\} \mid \leq n\}$

3.2 Attack Ontology

In order to extract the semantic information from raw alerts, the system's flaws and how the attacker breaches the network need to be known. For example, the "buffer overflow" flow can result in the attacker being able to run any arbitrary program and gain the root or administrator privileges. Attempts have been made to categorize and classify those computer attacks. The taxonomy should be specific and comply with the established security terminology. In [35], a summary of the computer misuse techniques was outlined as nine categories. The work in [36] develops a taxonomy of security threats: from the system to the program running on the system, then to the result of the breaches.

In Figure 3.1, John Howard [37] categorized the CERT incidents on the Internet and created an attack taxonomy with types of attackers, tools used, access information (such as why it was broken and what was used in the access), results of the break-in, and the objectives of the attack. The taxonomy of computer and network attacks developed for this research is used to present a summary of the relative frequency of various methods of operation and corrective actions. This taxonomy was taken and later expanded in the Sandia Laboratory [24].

The knowledge base refers to the representation of the relevant knowledge about the application domain. It contains the concepts, the instances, and the relations. In this section, $\mathcal{KB}$ is extended to the Attack Knowledge Base ($\mathcal{AKB}$) in the domain of intrusion security, which is an expressive modeling approach to implement SIM. $\mathcal{T}$ and $\mathcal{A}$ of $\mathcal{AKB}$, as well as the Attack Interpretation $\mathcal{AI}$, are defined so that $\mathcal{AI} \models \mathcal{AKB} =< \mathcal{T}, \mathcal{A} >$. In $\mathcal{T}$, semantic

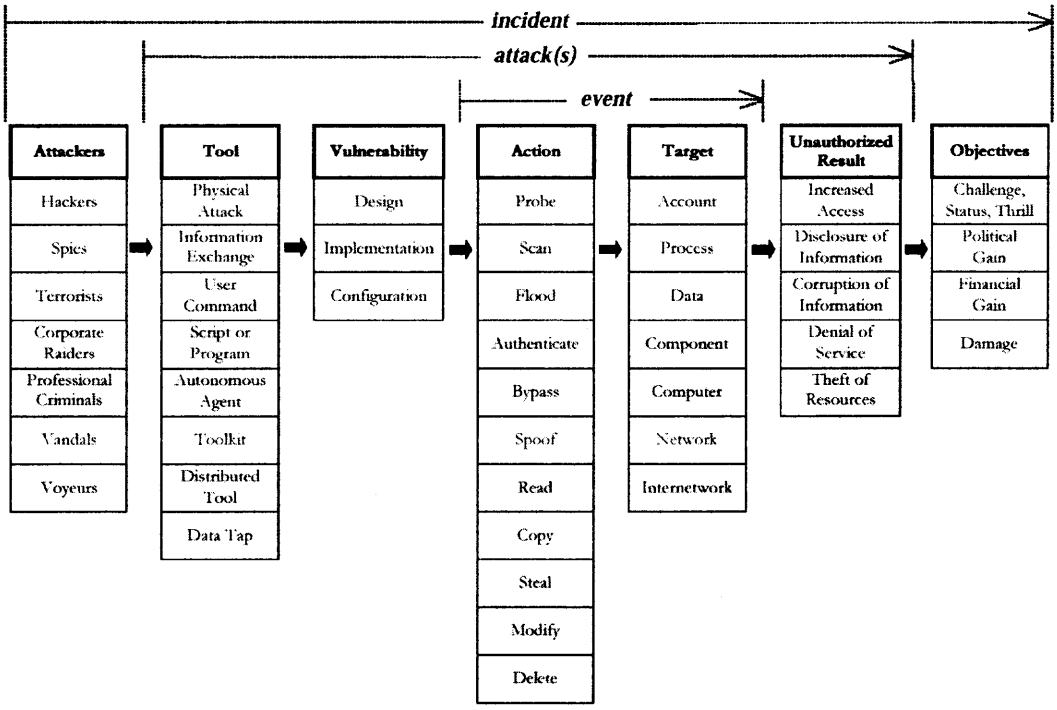


Figure 3.1 Howard's CERT Taxonomy [37].

intrusion attack ontology $\mathcal{O}_T$ is defined based on attack behavior semantics. In $\mathcal{O}_T$, the semantic roles chosen should reflect the semantic logic of attack actions. In this dissertation, $\mathcal{O}_T$'s classes are defined according to the interpretation of the noun sequences in the linguistic literature, and build up the sub-taxonomy for every class based on the CERT taxonomy.

3.2.1 Classification of Noun Sequences

Table 3.1 Jespersen's Classification Schema

RELATION NAME	EXAMPLES
Subject	earthquake, sunrise
Object	sun-worship, childbirth
Place	garden party, land-breeze
Time	daydream, wedding breakfast
Purpose	keyhole, dining room
Instrument	gunshot, footstep
Contained in	feather bed, sandpaper
Resembles	needle fish, silver-fox
Material	gold ring, stone wall

In NLP, theoretical linguistics has studied the classification of noun sequences (also known as noun compounds or complex nominals) from various perspectives. Most of these

Table 3.2 Lees' Classification Schema

RELATION NAME	EXAMPLES
Object/Property	collar size, vapor pressure
Whole/Part	oyster shell, whale bone
Contents	picture book, air pocket
Resemblance bulldog	hairspring
Form	brick cheese, lump sugar
Material	paper money, butter cookie

studies have focused on the semantic aspect, discovering which are the functional relations implicit between the nouns in a noun sequence. The discussions in Jespersen [38] and Lees [30] focused on providing a classification which is sufficient for the description of the noun sequences. The classifications in [38] are shown in Table 3.1, whereas Lees [39] mentioned several other categories shown in Table 3.2.

Table 3.3 Downing's Classification Schema

RELATION NAME	EXAMPLES
Whole-part	duck foot
Half-half	giraffe-cow
Part-whole	pendulum clock
Composition	Stone furniture
Comparison	pumpkin bus
Instrument	gunshot, footstep
Time	summer dust
Place	Eastern Oregon meal
Source	vulture shit
Product	Honey glands
User	flea wheelbarrow
Purpose	Hedge hatchet
Occupation	Coffee man

The description in Downing [40] focused on the functional roles of a noun sequence. This attack taxonomy was similar to Jespersen's because those classes were both descriptive in nature. Zachman's framework for information system architecture (ISA) [41] is widely used for developing an enterprise information systems architecture. The purpose of ISA is to provide a basic structure supporting the organization, integration, interpretation, development, and changing of a set of architectural representations of the organizations information systems. Tables 3.4 and 3.5 below represent the ISA's framework. The rows in Table 3.5 describe the views of the ISA participants, and the columns depict a different focus. (As Zachman put

it: "The same product can be described, for different purposes, in different ways, resulting in different types of descriptions") Together, these six interacting focuses describe the entire ISA architecture.

Table 3.4 Dimensions Summary

Dimension	Questions	House construction	Systems development
Entities	what	house, room	employee, department
Activities	how	play, eat	hire employee, promote employee
Locations	where	lot, rooms	headquarters, district office
People	who	occupants, guests, pet	human resources dept., recruiter
Time	when	construction sequence	during interview, each January
Motivations	why	reduce lawn maintenance	Ensure adequate staffing levels

Table 3.5 Questions Summary

Views	Object of focus	Focus items	Description
WHAT is it made of?	Data	Entities, relations	organizational information is made of data
HOW does it function?	Processes, functions	inputs, outputs	functional descriptions How does the organization do its work? How are orders filled? How is inventory maintained?
WHERE are things located?	Network	Nodes, links	where the work and information flows functional descriptions within the enterprise?
WHO is involved?	People	agent, work	people (employees) within the enterprise the work (or work products) they perform
WHEN do things happen?	time	time cycle	event-to-event relationships
WHY do things happen?	motivation	objectives, goals	the motivation of the enterprise

3.2.2 Attack Taxonomy in FAR-FAR

In this dissertation, the semantic roles are constrained based on the following questions that security administrators would naturally ask: *When did the actions happen? Where did the actions happen? By which means did the actions happen? What results did the actions cause? etc.,*

Figure 3.2 presents the three-layer $\mathcal{O}_T$ hierarchy of the concepts and relations. Each concept in the ontology is described by a set of attributes. Object role means the receiving end of the action, and it has *has object* and *be object of* attributes. The *meronymy* (has an object) and *holonymy* (is a part of) attributes from *part-whole* role describe the situations that one entity contains other entity. *Consequence tagging* role explains at which stage the attack

may locate: *gather information, making enable* or *launching attacks.* In $\mathcal{A}$, the instances about the concepts and semantic relations in $\mathcal{O}_T$ are stored in the intrusion attack instance base. The Attack Interpretation $\mathcal{AI} = (\triangle^{\mathcal{AI}}, \cdot^{\mathcal{AI}})$ represents the mapping form the behavior action space to attack scenario space. Every semantic role defined in $\mathcal{O}_T$ describes a semantic aspect of a certain action. Based on $\mathcal{O}_T$, $\mathcal{AI}$ maps these loose and uncorrelated individual actions into coherent attack plan by the semantic expressive description.

$\mathcal{O}_T = \{$

$$
\begin{aligned}
\textit{semantic roles} &\sqsubseteq \forall\ \textit{Location.Where} \sqcap \forall \textit{Method.What with} \sqcap \forall \textit{Object.Who/What} \\
&\quad \sqcap \forall (\textit{Possible}) \textit{Cause.What it causes} \forall \textit{Consequence tagging}, \\
\textit{Where} &\sqsubseteq \forall\ \textit{Include.Has location} \sqcap \forall \textit{Include.Be location of}, \\
\textit{What with} &\sqsubseteq \forall\ \textit{Include.Has instrument location} \sqcap \forall \textit{Include.By means of}, \\
\textit{Who/What} &\sqsubseteq \forall\ \textit{Include.Has object} \sqcap \forall \textit{Include.Be object of}, \\
\textit{What is part of} &\sqsubseteq \forall\ \textit{Include.Meronymy} \sqcap \forall \textit{Include.Holonymy}, \\
\textit{What it cause} &\sqsubseteq \forall\ \textit{Include.(Possible) Cause} \sqcap \forall \textit{Include.Be (Possible) caused by}, \\
\textit{Consequence tagging} &\sqsubseteq \forall\ \textit{Include.Gather Info} \sqcap \forall \textit{Include.Make enable} \forall \textit{Include.Launch attack}, \\
\textit{Has location} &\sqsubseteq \forall\ \textit{Has attribute.state=active} \sqcap \forall \textit{Has attribute.weight}=4, \\
\textit{Be location of} &\sqsubseteq \forall\ \textit{Has attributes.state=passive} \sqcap \forall \textit{Has attributes.weight}=4, \\
\textit{Has instrument} &\sqsubseteq \forall\ \textit{Has attributes.state=active} \sqcap \forall \textit{Has attributes.weight}=4, \\
\textit{By means of} &\sqsubseteq \forall\ \textit{Has attributes.state=passive} \sqcap \forall \textit{Has attributes.weight}=4, \\
\textit{Has object} &\sqsubseteq \forall\ \textit{Has attributes.state=active} \sqcap \forall \textit{Has attributes.weight}=2, \\
\textit{Be object of} &\sqsubseteq \forall\ \textit{Has attributes.state=passive} \sqcap \forall \textit{Has attributes.weight}=2, \\
\textit{Has location} &\sqsubseteq \forall\ \textit{Meronymy attributes.state=active} \sqcap \forall \textit{Has attributes.weight}=3, \\
\textit{Holonymy} &\sqsubseteq \forall\ \textit{Has attributes.state=passive} \sqcap \forall \textit{Has attributes.weight}=3, \\
\textit{(Possiblecause} &\sqsubseteq \forall\ \textit{Has attributes.state=active} \sqcap \forall \textit{Has attributes.weight}=3/5, \\
\textit{Be (possible)caused by} &\sqsubseteq \forall\ \textit{Has attributes.state=passive} \sqcap \forall \textit{Has attributes.weight}=3/5.
\end{aligned}
$$

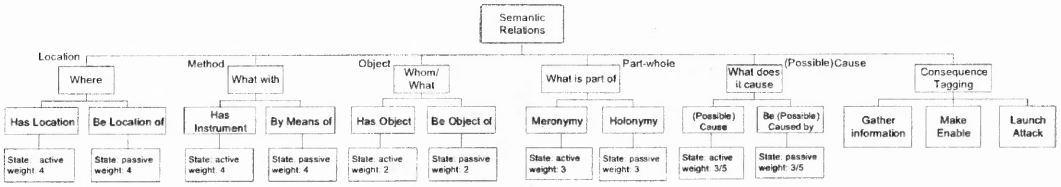


Figure 3.2 Ontology of semantic roles and attributes.

Definition 3.2.6 (*AKB* **Tbox** $\mathcal{T}$) *The* $\mathcal{T}$ *in* $\mathcal{O}_\mathcal{T}$ *is a 3-tuple* $\mathcal{T} =< C, R, A >$, *where* C *is a set of classes, which denote a set of the concepts,* R *is a set of the relations, which denote the binary relationships between the concepts, and* A *is a set of the concepts' attributes.*

Definition 3.2.7 (*AKB* **Abox** $\mathcal{A}$) $\mathcal{A}$ *contains the instances of* C *and* R *defined in* $\mathcal{O}_\mathcal{T}$, *and is 3-tuple consisting of concepts, relations, and instances,* $\mathcal{A} =< C, R, I >$, *where* I *is set of class attributes.*

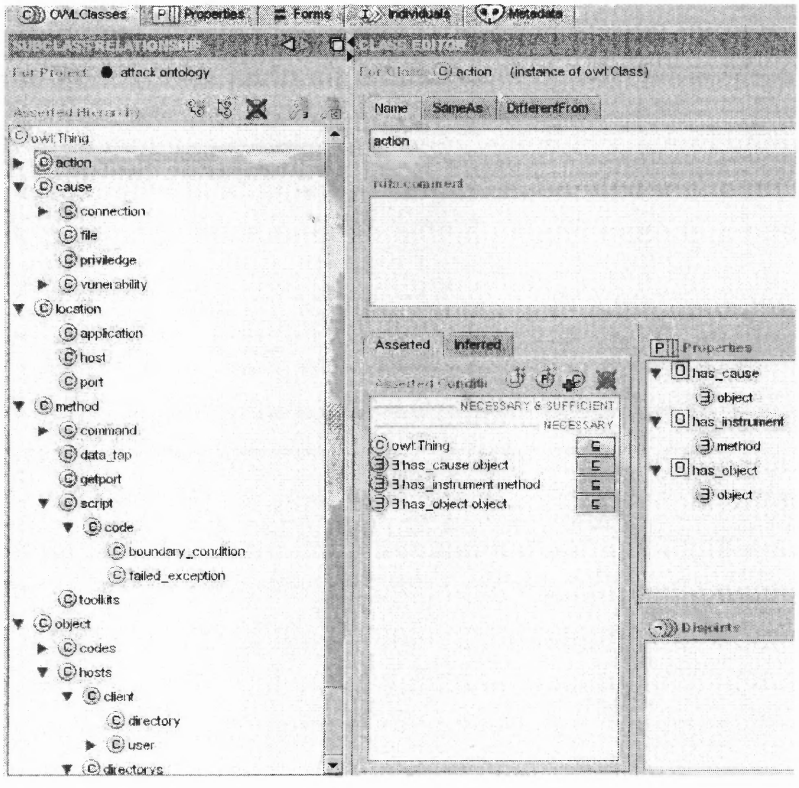


Figure 3.3 Example of attack taxonomy classes.

In FAR-FAR, the ontology is generated by protege [42], which can use a reasoner to check the consistency of the statements and definitions. FAR-FAR defines the classes and the properties. The class represents the object in the interested domain. For example, the empty ontology contains one class called $owl : Thing$, which is the class that represents the set containing all individuals. The properties are the binary relations on the classes. The

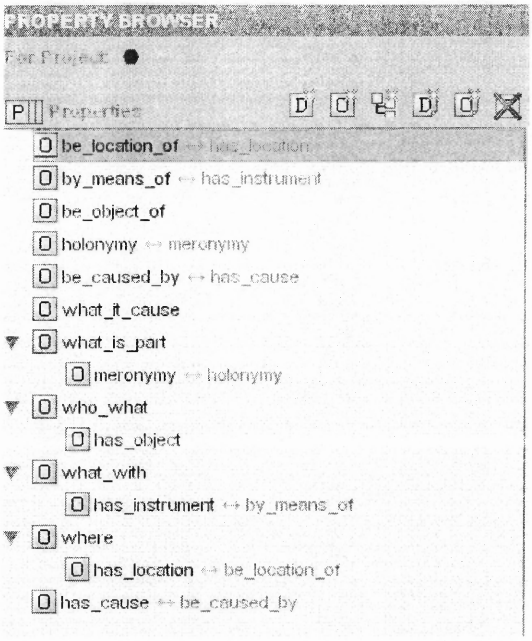


Figure 3.4 Example of attack taxonomy properties.

properties can have inverses. If some property links individual a to individual b then its inverse property will link b to a. For example, the inverse of *has object* is *be object of*. Figures 3.3 and 3.4 show the fragment of the classes and relations of the security taxonomy. Ontologies are also used in order to provide more precise semantics of the relationships between the terms and thus more precision may be applied when performing query expansion. Ontological information may also be used to help in query reformulation. If a query input does not have an answer, the query may be overly specific. Superclass may be used in order to make a query more general in an attempt to find answers.

3.3 PCTCG

Based on the definition of attack scenario in Section 1.3, the attack actions in the attack scenario can be converted into the semantic role structures which are independent of the specific attack events. As a matter of fact, linguistic methods are mature enough to acquire word semantics from the contexts. The formalism for actions used in most natural language understanding systems is based on case grammar. Each action is represented by a set of assertions about the semantic roles the noun phrases play with respect to the verb. Since every attack action can be regarded to be associated with a certain verb in linguistics, the

semantic information is extracted from heterogeneous alerts by the modified Case Grammar theory, PCTCG [43].

The reasons for using Case Grammar are three folds: First, Case Grammar structure specifies the semantic relations (slots) between a verb and other entities [44]. Frame is useful in representing descriptive or stereotypical conceptualization of events. Second, Case Grammar is easily represented by a semantic network, which includes abundant semantic relations that can express the alerts' associations. Third, unlike the syntactic level, Case Grammar theory is deep semantic [45], which means it does not change under grammatical transformations. As shown in Table 3.6, *Subject* role and *Object* role change in syntactic level when the sentence switches from active to passive form. However, the *Agent* (*Agent* is what causes the action of the verb) and *Theme* (*Theme* is the object in motion or being located) roles in the semantic level remains the same.

Table 3.6 Syntactic vs. Semantic

Sentence	Syntactic level		Semantic level	
	Object	Subject	Agent	Theme
The man moved the desk.	the man	the desk	the man	the desk
The desk was moved by the man.	the desk	the man	the man	the desk

The aim of PCTCG is to normalize the aggregated intrusion alerts into uniform semantic representation of attack behavioral actions. An attack scenario can be regarded as a sequence of attack events, in which each includes a certain attack action. When considering two alerts (two actions), the semantic roles need to be used to correlate them, and the *Principal-subordinate* relation is applied on the two alerts. When one alert is in the *principal* phase, it is treated as a verb and replace the other alert with its subordinate keywords (noun phrases). If the subordinate keywords arein a specific case relationship with the verb, then these two alerts are correlated.

PCTCG is formally defined as $G = \{M_n, C, F, S\}$, where M_n is the alert messages set of the IDS sensor with sensor name n, C specifies the set of possible semantic roles (slots) between alerts, F is the set of case fillers (legal value for each slot), and S is the set of subordinate keywords. Traditional case grammar includes a selection restriction, which

specifies accepting or rejecting case fillers between the main verb and the noun phrases [45] (for example, the object of the verb drinking must be a liquid substance).

In PCTCG, since the selection restriction of case grammar has been shifted to the semantic knowledge databases, the set of selection restrictions will not be included. For every alert, there are several subordinate keywords which can describe the alert background well. For example, consider two Snort alerts: *FINGER 0 query* and *FINGER redirection attempt.* Their semantic attributes and case fillers are shown in Table 3.7.

Table 3.7 Semantic Attributes and Case Fillers

FINGER 0 query	
Semantic roles	Semantic case fillers
Has object	FINGER daemon
Possible cause	User account, password
By means of	FINGER command with username '0'
Consequence tagging	Make enabling
Subordinate keywords	User name, FINGER daemon
FINGER redirection attempt	
Semantic roles	Semantic case fillers
Has object	FINGER requery
Possible cause	Gain information
Cause	DDoS, indirect connection
Consequence tagging	Launching attack
Subordinate keywords	FINGER requery, third party

In this dissertation, FOL [46] is used as the alert representing and reasoning language. In FOL, there are three types of logic symbols:

- punctuation: "(", ")", and "."
- connectives: "$\neg$", "$\forall$", "$\exists$", "$\vee$", "$\wedge$", *and* "$\equiv$". Note that "$\neg$" is logical negation, "$\wedge$" is logical conjunction, "$\vee$" is logical disjunction, "$\forall$" means "for all...", "$\exists$" means "there exists...".
- predicate: predicates denote the semantic roles defined in $\mathcal{O}_{\mathcal{T}}$.

Based on the alert semantic information, their PCTCG streams are represented by predicate logic format:

$$E[M_n{:}(FINGER\ 0\ query)_{snort}]=$$

$$\exists e.[\exists v[command(C{::}has\ object(FINGER\ daemon),third\ party,v)]\wedge C{::}possible\ cause$$

$$(User\ account,\ password)\wedge C{::}cause\ (FINGER\ command\ with\ username\ '0')$$

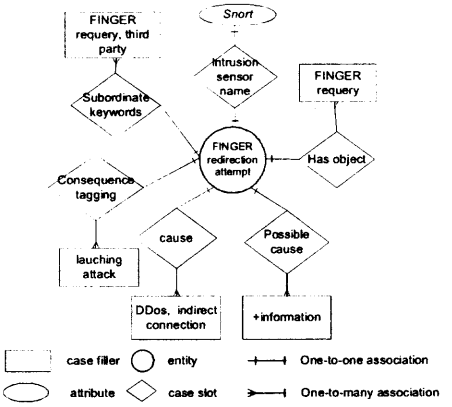


Figure 3.5 PCTCG diagram.

$\wedge C$::*consequence tagging*(*launching attack*)$\wedge S$:(*Finger query, third party*)].

$E[M_n$:(*FINGER redirection attempt*)$_{snort}]$=

$\lambda e.[\exists v[forward(C$::*has object*(*FINGER query*),*third party*,$v)]\wedge C$::*possible cause* (*gain info*)

$\wedge C$::*cause* (*DDos, indirect connection*)$\wedge S$:(*Finger query, third party*)].

where E is an entity described as "the event in which the Finger daemon forward the query to the third party". $\exists e$ means "there exists an event...", $\exists v$ means "there exists a kind of attack action...", " $\wedge$ " is logical conjunction, and :: means "include". Here, *has object, possible cause, cause, consequence tagging* are the semantic roles, *finger requery, +info,DDoS, indirect connection, launching attack* fill the slots of the above roles respectively, and *FINGER requery* and *thirty party* are the subordinate keywords. Predict logic describe the conjunction of the action predicate with other predicates described in the event. Based on [34], the following formulas are defined to translate the predict logic into DL.

Definition 3.3.8 (Predict logic transform axioms) *The following axioms are defined to translate the predict logic to DL:*

- $\vartheta_{\exists C} = \exists x \bigwedge C(x)$
- $\vartheta_{\exists R.C}(x) = \exists y.R(x,y) \bigwedge C(y)$
- $\vartheta_{\forall R.C}(x) = \forall y.R(x,y) \bigwedge C(y)$
- $\vartheta_{\leq nR}(x) = \forall_{y_1 \cdots y_i \cdots y_n}(y) R(x,y_1) \bigwedge \cdots \bigwedge R(x,y_{n+1}) \rightarrow \bigvee_{i<y}(y_i) = y_j$
- $\vartheta_{\geq nR}(x) = \exists_{y_1 \cdots y_i \cdots y_n}(y) R(x,y_1) \bigwedge \cdots \bigwedge R(x,y_{n+1}) \rightarrow \bigwedge_{i<y}(y_i) \neq y_j$

In Figure 3.5, the Entity-Relationship (E-R) diagram is used to represent the PCTCG format. The E-R scheme can be translated into DL description. Each entity in the E-R diagram can be translated into a concept in DL, while each E-R role is translated into a DL role. The DL statement of E-R can be expressed as follows:

$$
\begin{aligned}
&(\textit{Finger redirection attempt})_{Snort}=\{\\
&\quad \vartheta_{\exists INTRUSION\ SENSOR\ NAME}Snort,\\
&\quad \cap\vartheta_{\exists HAS\ OBJECT.Finger\ query}\textit{Finger redirection attempt},\\
&\quad \cap\vartheta_{\exists POSSIBLE\ CAUSE.gain\ information}\textit{Finger redirection attempt},\\
&\quad \cap\vartheta_{\exists CAUSE.DDOS,\ indirect\ connection}\textit{Finger redirection attempt},\\
&\quad \cap\vartheta_{\exists CONSEQUENCE\ TAGGING.launch\ attack}\textit{Finger redirection attempt},\\
&\quad \cap\vartheta_{\exists SUBORDIANATE\ KEYWORD.Finger\ query,\ third\ party}\textit{Finger redirection attempt}\}.
\end{aligned}
$$

3.4 FOL Backward-chaining Reasoning

A clause is interpreted as the disjunction of a set of literals. The clause: $B_1, \cdots, B_n \Rightarrow A$, if B_i is true for $i = 1, \cdots, n$, then A is true. A statement is called a Horn clause if it contains only at most one negative literal [46]. For example, a *positive* Horn clause $[x_1, x_2, \neg y]$ can be thought as $x_1 \wedge x_2 \Rightarrow y$ or "if both x_1 and x_2 happen, then y must happen".

Backward or forward chaining are two primary methods for knowledge reasoning. Backward chaining starts with the hypothesis, and it works backward from the hypothesis to the facts which support the hypothesis. Forward chaining travels from the facts to the conclusions. To determine whether backward or forward chaining should be used depends on the specific problems. If the given attack facts can reach a large number of resolutions, and few of which you are interested, then backward chaining should be adopted. Otherwise, forward chaining should be used. To determine which method is better in reasoning the attack scenarios, consider the following situation. For the security administrator, what are known to him/her are the vulnerabilities of the networks and hosts, the malicious attacks being happened, and the generated alerts; he/she also wants to know if some specific attack attempts happened. Equivalently, in the view of SIM, FAR-FAR stores the vulnerabilities of the networks and hosts into the AKB in advance, and converts the generated alerts into

frame-based streams, and then into Horn clause facts. To extract the attack scenarios, FAR-FAR needs to check whether or not the AKB together with the production rules can satisfy questions related attack attempts: when, where, by which means,what resulted from the attack? With the semantic roles defined in $\mathcal{O}_{\mathcal{T}}$, the reasoning goals can be pre-defined. Furthermore, to guarantee real-time efficiency, reasoning should be done without unrelated conclusions. Therefore, backward chaining is chosen; that is, given some production rules and a AKB containing a set of FOL Horn clauses, whether or not the interested attack attempts can be derived needs to be known.

In the simulations, Prolog is used as the backward chaining language. By backtracking, the inference engine provides Prolog with powerful reasoning capabilities [47]. A Prolog program consists of a sequence of goals G, which are either atomic goals or negations of atomic goals, the production rules R, and a set of predicate definitions P, which are either alert's frame structures or the ontology attack facts. The production rules are expressed as conditional sentences in the form:

```
IF semantic roles match THEN semantic correlation.
```

For example, the attack facts and *object* correlation rule of alert *FINGER requery* can be represented in Prolog as clauses:

1. `object("finger_requery", "finger_daemon").`
2. `method("user_account", "finger_requery").`
3. `idsEvent("finger_0_query", "finger_daemon", "finger_daemon", "empty", "user_account", "empty", "empty", "finger_with_username_0" ).`
4. `judge_object(PAlert, SlotFiller, SAlert, Keyword, SlotName) :-`
5. `object(SlotFiller, Keyword).`

3.5 First-order Logic Reasoning for Alert Correlation

In this section, FAR-FAR's automatic alert correlation reasoning will be expounded in detail. Here, reasoning refers to the manipulation of the attack facts stored in AKB to generate the logic conclusion of the attack scenario. Given the PCTCG frames, the semantic operatoins and the intrusion database containing the attack knowledge facts, whether or not the goals of interested correlation can be automatically derived needs to be known.

The reasoning process has two steps: Conjunctive Normal Form (CNF) formalization, and FOL resolution. In the CNF formalization, PCTCG frames and the attack knowledge facts are all formatted into CNF clauses. In the reasoning process based on [46], And-Or Resolution Tree is proposed to derive the alert correlations from CNF clauses automatically using the substitution and the semantic fusion. Substitution is to replace an alert with its subordinate keyword, whereas the semantic fusion is the process of fusing two clauses into one.

Definition 3.5.1 *CNF is a conjunctive of disjunctions of literals.*

For example, a CNF has the following form:

$$(x_1 \vee x_2) \wedge \cdots \wedge (x_{n-1} \vee x_n)$$

where $x_i, 1 \leq i \leq n$ is a propositional logic clause.

Because alerts' PCTCG streams are unit clauses in conjunction format, they are already in CNF format. For example, CNF of alert 1 *FINGER 0 query* and alert 2 *FINGER redirection attempt* are:

CNF(alert1, alert2)=

[has object(alert1, FINGER daemon)

∧ possible cause(alert1, User account)

∧cause(alert1y, FINGER command with username '0')

∧S(alert1, Finger query)]

∧[has object(alert2, FINGER query)∧possible cause (alert2, gain info)

∧cause (alert2,DDos & indirect connection)

∧S:(alert2, Finger query & third party)].

In FAR-FAR, the intrusion database stores the attack knowledge facts, which are the unit CNF clause describing a certain semantic role. For example, the facts related with the alert *FINGER requery* are *object(finger_requery, finger_daemon)* and the *method(user_account, finger_requery)*. For the reasoning questions, for example, "if both x_1 and x_2 exist, then must y happen?" can be thought as $x_1 \wedge x_2 \Rightarrow y$ and expressed as the CNF clause $x_1 \wedge x_2 \wedge \neg y$.

Definition 3.5.2 *For two literals, l and l', they are called the complement literals if $l' = \neg l$.*

Definition 3.5.3 *Consider two CNF clauses $c_1 \cup \{l\}$ containing the literal l, and $c_2 \cup \{l'\}$ containing the complement of l, resolution is the process of inferring the clause $\{c_1 \cup c_2\}$ based on the complement literals l and l'. In this case, $c_1 \cup \{l\}$ and $c_2 \cup \{l'\}$ are resolvent. $c_1 \cup \{l\}$ and $c_2 \cup \{l'\}$ are parent clauses and $\{c_1 \cup c_2\}$ is the child clause. The literals l and l' are called complete resolvent.*

Definition 3.5.4 *The alert correlation goal G consists of the subgoal list defined in the attack ontology, $G = \{g_{object}, g_{spatial}, g_{instrument}, g_{part-whole}, g_{cause}\}$. If one or more subgoals are satisfied, the whole goal G is successful.*

Definition 3.5.5 *A substitution Θ is an element from a set of pairs $\{x_1/t_1, \ldots, x_n/t_n\}$, where $x_i, 1 \leq i \leq n$, is the subordinate keyword of alert t_i, and x_i/t_i means that subordinate keyword term x_i is substituted by its alert t_i throughout the resolution.*

Definition 3.5.6 *There exist goal $G = \{g_{object}, g_{spatial}, g_{instrument}, g_{part-whole}, g_{cause}\}$, and a set of clauses $C = (c_1, c_2, \cdots, c_i, \cdots, c_m)$ where $1 \leq i \leq m$. Assume there is a substitution θ_i so that $h_i = \theta_i c_i$. Let $C' = C\theta_i = (c_1, c_2, \cdots, c_j, \cdots, c_m)\theta_i = \{\theta_i c_1, \theta_i c_2, \cdots, h_i, \cdots, \theta_i c_m\}$. Then C' is called the substitution step with input clause C and θ_i, and $C' = \theta_i(C)$.*

Definition 3.5.7 *Suppose two clauses $R_1(c_i, c_j)$, and $R_2(c_j, c_k)$ from a set of clauses $C = (c_1, c_2, c_i, \cdots, c_j, \cdots, c_k. \cdots, c_m)$ where $1 \leq i, j, k \leq m$, and $i < j < k$. Assume $R_1(c_i, c_j) \star R_2(c_j, c_k) = R_1 \star R_2(c_i, c_k) = R_3(c_i, c_k)$ where $\star$ is the semantic operation, and $R_1 \star R_2 = R_3$ means that the semantic operation R_1 and R_2 can be fused into R_3. Let $C'' = Cf_{(R_1 \star R_2 = R_3)} = (c_1, c_2, \cdots, c_{j-1}, c_{j+1}, \cdots, c_m)$. Then C'' is called the semantic fusion step with input clause C and $f_{(R_1 \star R_2 = R_3)}$, and $C'' = f_{(R_1 \star R_2 = R_3)}(C)$.*

Definition 3.5.8 *Suppose there exists $C = \{Alert1, Alert2, x_1, x_2\}$ that $R_2(x_1, x_2)$, $R_1(Alert1, x_1)$, and $S(Alert2, x_2)$. And C has a semantic fusion step $f_{(R_1 \star R_2 = R_3)}$ and a substitution step $\theta_i = (x_2/Alert2)$, then*

$$R_1(Alert1, x_1) \wedge S(Alert2, x_2) \models R_3(Alert1, Alert2)$$

where $\models$ is called a FOL resolution step.

Definition 3.5.8 *A FOL attack reasoning derivation of goal G from a set of clause C is a finite clause sequence: $C_1, \cdots, C_n \Rightarrow G, 1 \leq i \leq n$ such that each clause C_i is the resolution step of C_{i-1}, which is written as $C \models G$. A reasoning derivation is successful if there exists $c \in C_i$ where c and the subgoal $g' \in G$ is complete resolvent (that is, $g' \wedge c = \emptyset$); it is unsuccessful if the last resolution is non-empty, but no further resolution step is possible.*

Definition 3.5.9 *A successful resolution tree of $C \models G$ is a labeled binary tree such that the root is labeled $\emptyset$, the leaves are labeled with elements of C, and each nonleaf is labeled with a resolvent of the labels of its immediate successors. The root is at the bottom. For a set of clauses, there exists at least one alert correlation if there exists a successful resolution tree.*

To decide if the goal $G = \{g_{object}, g_{sptial}, g_{instrument}, g_{part-whole}, g_{cause}\}$ can be derived from the clauses, FAR-FAR tries to completely resolve the left-most subgoal with the clause which has the same predicate name. When the resolution is successful, FAR-FAR attempts another resolution step. When the resolution fails, the process backtracks and tries another inference step. The whole procedure constitutes the depth-first and left-to-right order. The attack resolution derivation is commonly represented by a binary reasoning tree, drawn with its root at the bottom. Resolution tree is a data structure used for building binary resolution proofs in FOL. The main focus is finding efficient ways to extract the correlation between alerts. All nodes are labeled by the literals which are still to be resolved (known as the clause label), while internal nodes are labeled with the letter of the atom being resolved. Consider an automated reasoning procedure that takes a set of clauses as input, and outputs the correlations.

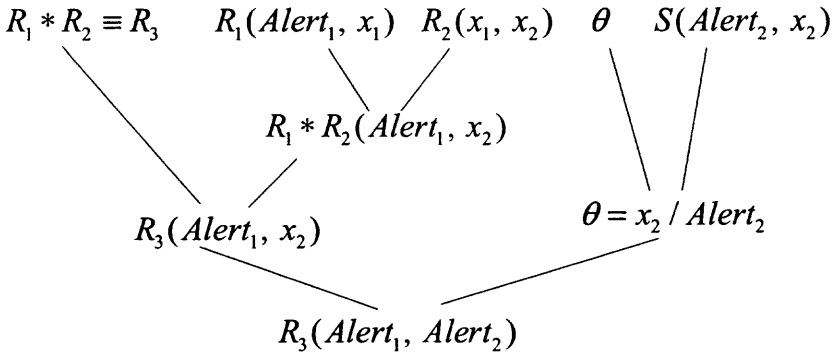


Figure 3.6 First-order attack reasoning tree.

Figure 3.6 shows the process of the resolution tree between alert1 and alert2. Initially, there are no connections between the case filler variables. When performing resolution, the variables are renamed so that they have no variables in common. Afterwards, the resolution tree tries to determine if the semantic roles R_1 and R_2 can be fused into R_3. Here, $\star$ is the semantic role operator. If there exists a semantic operation $R_1 \star R_2 \equiv R_3$ and a semantic role fact $R_2(x_1, x_2)$, which means x_1 has a semantic role R_2 with case filler x_2, the resolution of $R_1(Alert1, x_1)$ and $S(Alert2, x_2)$ is defined as follow:

$$R_1(Alert1, x_1) \wedge R_2(x_1, x_2) \models R_3(Alert1, Alert2)$$

where $R_1 \star R_2 \equiv R_3$ means that semantic roles $R1$ and $R2$ can be fused into $R3$, $S(Alert2,\ x_2)$ means that $Alert2$ has subordinate keyword x_2, and the substitution Θ is instantiated with $x_2/Alert2$.

3.6 Link Padding

Preventing networks from being attacked has become a critical issue for network administrators and researchers. As a precursor to a network attack, attackers may perform traffic analysis whereby the aim is to derive mission critical information based on an analysis of the traffic over the network. One can defend against traffic analysis attacks with the encryption, such as node encryption or link encryption. Although with encryption, the attacker will not be able to decode the context of the packets. However, the network links between the nodes can still be vulnerable to traffic analysis attacks. For instance, using the "packet counting" attack, the attacker can count the number of packets entering and leaving one node, and determine the next node which the packets will be sent. With the gained link information, the attacker can launch DDoS on the nodes.

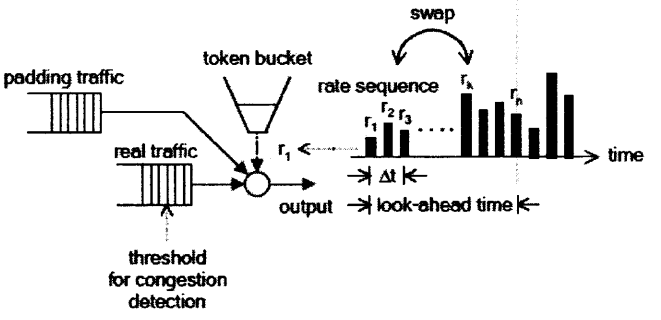


Figure 3.7 Link padding method in [2].

Another countermeasure against traffic analysis is to use link padding where the cover traffic and the real traffic are mixed so that every link's total traffic looks constant or similar to the attackers. The clients can then transmit a payload independent stream of data on the links to the servers. In [48], the variant interval link padding method was compared with the constant interval link padding method. The results indicated that the constant interval link padding may fail in preventing traffic analysis from determining the rate of real payload traffic, while the variant interval link padding based countermeasures seem to be effective. A method of constant interval link padding was described in [2], which is shown in Figure

3.7. This method chose a value from a heavy tail distribution as the current traffic rate and swap with a more suitable value if the current value does not meet the link traffic rate requirement. However, it has several drawbacks: First, it requires all paths to generate the cover traffic for every requesting path. Second, since the solution is constant link padding, the densely incoming packets may subtly delay the timer's interrupt routine which is in charge of generating the cover traffic and the attacker can analyze the timer's delayed time to obtain the link information [48]. Furthermore, in the heavy tail distribution, the probabilities of the larger values are much smaller than the smaller ones. Most traffic rate candidates are very small that does not meet for the requirement of high speed link transmission.

Yan et al. [49] proposed the Variant Packet Sending-interval Link Padding method (VPSLP) to defend against the traffic analysis attacks on the links between the IDS sensors and FAR-FAR based on the Pareto distribution. The Pareto distribution is a simple heavy tailed distribution with the probability mass function defined as: $P[x] = \alpha k^{\alpha} x^{-\alpha-1}$ where $\alpha, k > 0,\ x \geq k$. Figure 3.8 shows a simplified anonymity system including IDS sensors and agents. Link encryption is not enough to protect the anonymity of the system. For example, if the attacker needs to know which sensor A is communicating with agent A. The attacker can eavesdrop the links before and after the ingress anonymizing node, and collects the number of the packets transmitted or the packet inter-arrival time. After the comparisons, the attacker can derive the route between sensor A and agent A, and launch attacks on the nodes. The countermeasure against traffic analysis is to use link padding where the cover traffic and the real traffic are mixed so that every link's total traffic looks constant or similar to the attackers.

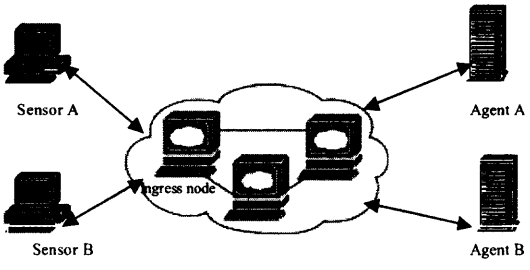


Figure 3.8 Anonymous links in FAR-FAR.

As shown in Figure 3.9, consider the link between a client and a server. On the client side, there exist two kinds of buffers: the traffic buffer and the constant length buffer. The function of the traffic buffer is to store the incoming packets. Suppose the traffic buffer is

large enough so that it will not overflow (since the link transmission rate in the anonyminity systems is not very high). The constant length buffer sends the packets exactly according to the generated heavy tail distribution. Every value from the heavy tail distribution can be treated as a timer. If the timer does not expire, the constant length buffer will hold the packet, otherwise, the packet is sent right away. The swapping between the values of the heavy tail distribution is not allowed. Let the constant length buffer's length be l. If the incoming packet's length is bigger than l, the sending anonymizing node can split the packet into several segments and the receiving anonymizing node can combine the segments. Note that the smallest value in the heavy tail distribution is larger than the time to fill up the constant length buffer, and the time to pad the whole constant length buffer can be ignored. When the constant length buffer is filled up, it sends out the packet and fetches the next value from the heavy tail distribution. The time to fetch the value from the distribution is also considered negligible. The cover traffic is generated under two conditions. First, the sum of the incoming packet size and the total size of the packets in the buffer is greater than l, then the cover traffic is padded to fill up the constant length buffer. If the timer does not expire, the buffer will hold until the timer expires. Otherwise, the buffer sends the packet out immediately. Second, if the constant length buffer is not padded up and the timer has already expired, the cover traffic is padded to fill up the buffer, and then the packet is sent out.

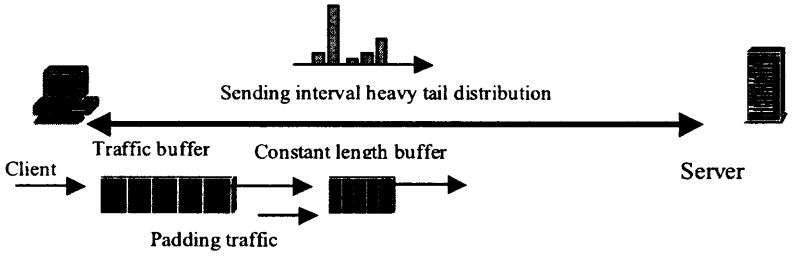


Figure 3.9 Variant Sending-interval Link Padding.

VPSLP does not generate the cover traffic all the time, but based on the incoming traffic and the generated heavy tail distribution. VPSLP is also better than the method which inserts the cover traffic at random, because the randomness can often be removed by using the statistical methods. Links with the same starting node are called the node's link group. Every link group uses the same value of a and k to generate the heavy-tail distribution. If

the control unit detects the traffic change on any link within the link group, lets $l = \mu$, and $k = \mu - \gamma$. Initially, the three Pareto distributions ($\alpha = 1.3, 1.5$, and 2.0) are mixed with equal probability. They are modified based on the change of the traffic burstiness degree. Let $< A_1, A_2, A_3 >=< 0.75, 0.85, 1 >$ and P_1, P_2, and P_3 be mixed probabilities of the link0, link1, and link2, respectively,

$$P_i = \begin{cases} \frac{H}{A_i}, & i = j \\ \frac{A_i - H}{2A_i}, & i \neq j \end{cases} \quad i = 1, 2, 3. \qquad j = \begin{cases} 1, & 0.5 \leq H < 0.75 \\ 2, & 0.75 \leq H < 0.85 \\ 3, & 0.85 \leq H \end{cases} \tag{3.1}$$

The packet sending-interval distribution of the whole link group will change simultaneously to defend against the traffic analysis attack.

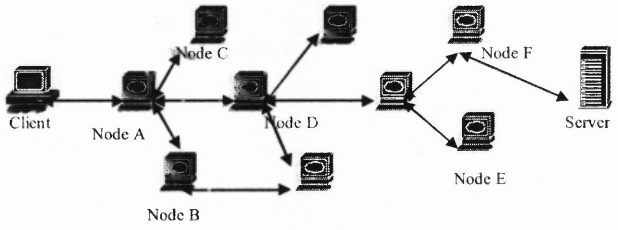
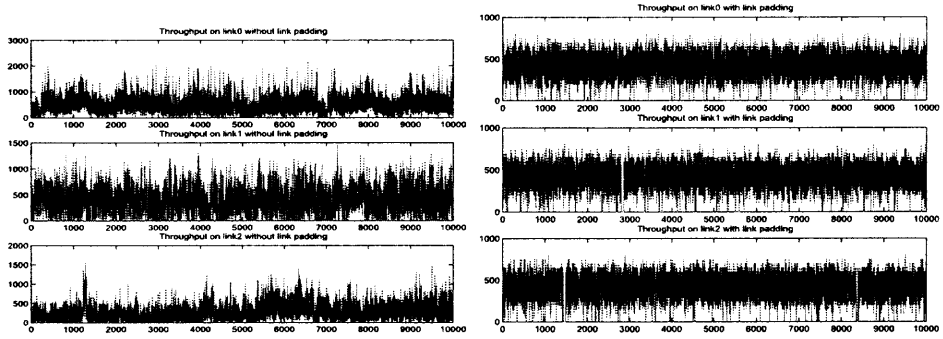
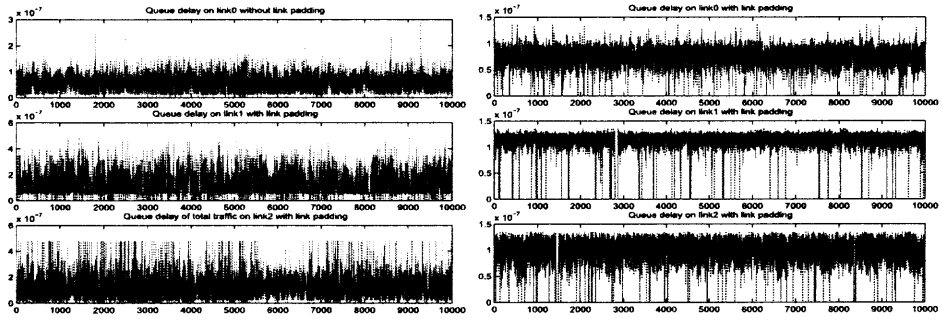


Figure 3.10 The scenario of the VPSLP.

In IDS, the anomaly detection is often used to find the attack traffic. Consider the scenario shown in Figure 3.10, after compromising the client, the attacker can pour the abnormal traffic, whose destination is to the server, into the link between client A and the ingress anonymizing node of A. The abnormal data may have specific pattern of the packet interval time or the packet size. Without the link padding, the abnormal pattern will be present on the link between node A and node D (since A $\rightarrow$ D is part of the route to the server). In VPSLP, because the control unit adjusts only the sending-interval heavy tail distribution and l, based on the traffic changes, the abnormal traffic can only have effect on the heavy tail distribution and l, and none of the three links (node A to node B, node A to node C, and node A to node D) will have the abnormality. However, if we apply the traffic anomaly detection on the control unit, the abnormal traffic can be detected.



(a) Throughput before VPSLP. (b) PCTCG Throughput after VPSLP.



(c) Queue delay before VPSLP. (d) Queue delay after VPSLP.

Figure 3.11 Link padding simulation results.

We simulated three links: node A to node B, node A to node C, and node A to node D, which were labeled as link0, link1, and link2 respectively. Traffic datasets from [50] are used. As seen in Figure 3.11, without link padding, the throughputs of all three links in 200 seconds are very much different due to the different input traffic traces. With link padding, the traffic throughput patterns of all the links are statistically similar.

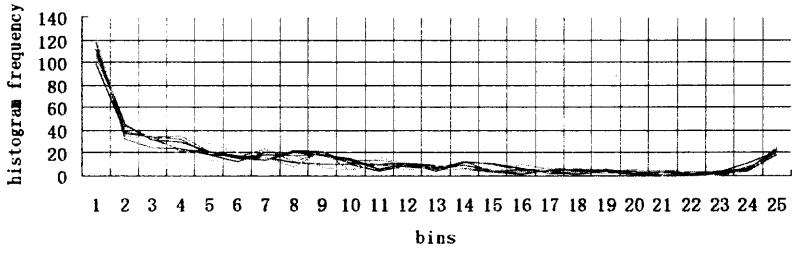


Figure 3.12 HFVs of traffic after link padding.

As shown in Table 3.8 and Figure 3.12, unlike the original traffic without link padding, the descriptive statistics of link traffic with link padding and Histogram Feature Vector (HFV) are statistically similar. Furthermore, the performance of the end-to-end delay of the real traffic in one second does not degraded significantly as compared with the original traffic.

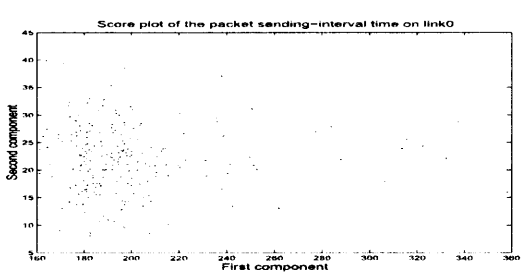
Table 3.8 Descriptive Statistics of Link Traffic with Link Padding

Link	Mean	StDev	Q1	Q3	Entropy
before link padding					
Link 0	492	295	300	650	-
Link 1	379	243	200	550	-
Link 2	259	200	100	350	-
after link padding					
Link 0	423	136	350	500	8.978
Link 1	417	141	350	500	8.926
Link 2	375	132	300	450	8.802

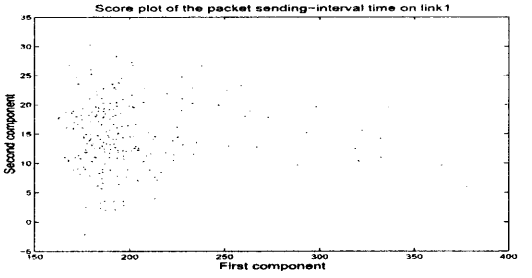
This is because the generated heavy tail distribution fit well with the original traffic's packet inter-arrival time distribution.

To defend against the attacker's abnormal traffic, the work in this dissertation takes advantage of the histogram skewness distribution and uses the histogram feature vectors as the input features to the Principle Component Analysis (PCA). On the anonymizing node, the packet inter-arrival time series is divided into certain segments (for example, 400 values every segment). Then, every segment is divided into k bins and compute the histogram of k equivalent time bins for each segment. The scope chosen for the histogram in each data set is $1.0\mu \sim 2.0\mu$ from the mean. In the simulation, $k = 14$ is chosen. The reasons to use PCA are following: 1) In the network traffic, the bins are a set of correlated variables. PCA can transform them into a smaller set of uncorrelated variables. 2) High dimension requires high computation cost, so PCA is used to reduce the high dimension to two dimension. 3) PCA can minimize the average projection errors [51]. The main use of PCA is to reduce the dimensionality of a data set while retaining as much information as possible. PCA is a multivariate procedure, in which a set of correlated variables are transformed into a set of uncorrelated variables. PCA chooses orthogonal linear combinations of the predictor variables that maximize the variance. PCA allows us to visualize and analyze the M observations (initially described by the N variables) on a low dimensional map. PCA provides the optimal view for a variability criterion, and builds a set of P uncorrelated factors ($p \leq N$) that can be reused as input for other statistical methods [52].

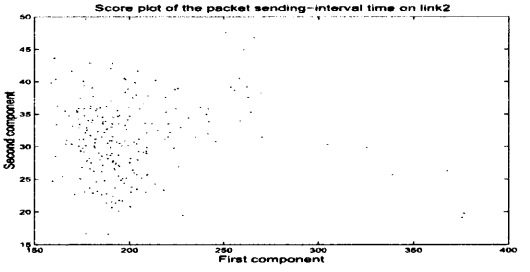
The total number of anomaly trace segments is 359 traces, while the normal ones are 646 traces. The histogram trace generated is used as a feature vector. Then, PCA is applied



(a) Link 0 PCA.

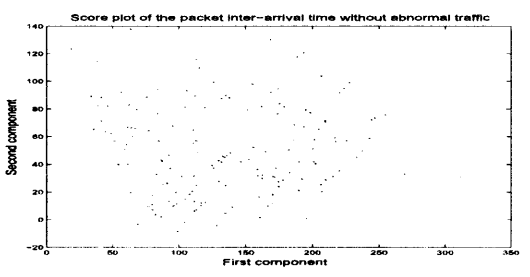


(b) Link 1 PCA.

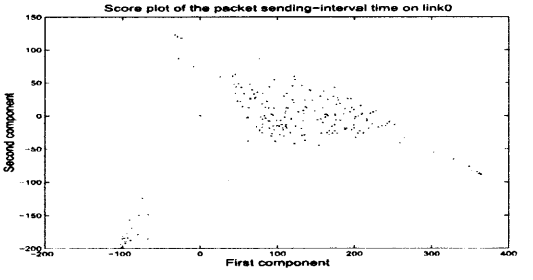


(c) Link 2 PCA.

Figure 3.13 PCA results of packet inter-arrival time.



(a) PCA results before inserting abnormal traces.



(b) PCA results after inserting abnormal traces.

Figure 3.14 PCA results before and after inserting abnormal traces.

on the vector and projects it into the first two components. The histogram feature vectors generated above are used as the input to PCA, which projects the results into the first two components. Figure 3.13 shows the PCA results of the three incoming packet inter-arrival time series of node B, node C, and node D. If the attacker inserts abnormal traffic to a node, it can be detected by applying PCA and the node's control unit will not adjust the heavy tail distribution and l, thus defeating the attack. Figure 3.14 are the PCA results before and after adding the abnormal traces. The abnormal traffic is mapped to the left corner.

CHAPTER 4

FIRST-ORDER REASONING

In this chapter, FAR-FAR's automatic alert correlation reasoning will be expounded in detail. Given PCTCG frames, the semantic operations rules, and the intrusion database containing the attack knowledge facts, the security administrator wants to know whether or not the goals of interested correlation can be automatically derived.

4.1 And-Or Correlation Rule Tree

To extract the alert correlations, an And-Or Correlation Rule Tree (AOCRT) is proposed. AOCRT includes the correlation rules, the attack facts (attack knowledge facts), and the semantic operations. Using the PCTCG streams as inputs, AOCRT is traversed to produce the attack scenarios as the outputs. As shown in Figure 4.1, AOCRT is divided in two parts: the semantic matching and the resolution tree. In semantic matching, 2-AASN is used to correlate two alerts; that is, the *principal* $\rightarrow$ *subordinate* correlation and *subordinate* $\rightarrow$ *principle* correlation. Afterwards, the FOL resolution tree tries to determine if the semantic matchings satisfy the following correlation rules: *cause, location, instrument, object, part-whole*, and *possible cause*, by checking the semantic role facts stored in the attack database and semantic operation facts.

4.1.1 2-Atom Alert Semantic Network

Semantic networks have been used in artificial intelligence for representing knowledge. It is a directed graph used for representing objects and the relations among them. The objects, shown as labeled circles, are referred to as nodes. The properties and relations of objects are expressed as directed arrows connecting the nodes. In this dissertation, 2-AASN is proposed as the semantic correlation representation between two alerts based on [53]. The edges of 2-AASN represents PCTCG semantic attributes or the label subordinates, and the nodes represent two atom alerts or their child nodes: case filler or the subordinate keyword. The formal description of 2-AASN is based on the 2-tuple slot, $<$ *semantic attributes*, *case*

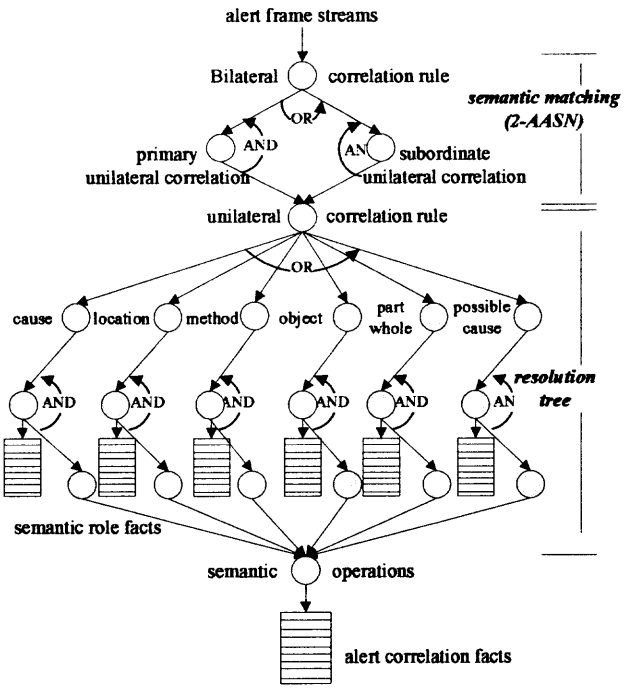


Figure 4.1 And-Or correlation rule tree.

filler >, or < *subordinate, subordinate keyword* >, which describes the semantic role or subordinated keyword:

$SN[node1,node2]=\{$

$node1:<subordinate,node1::subordinate\ keyword>^{+)}$,

$node2:<semantic\ attributes,\ node2::case\ filler>^{+)}$,

$node2::case\ filler<semantic\ attributes,\ node1::subordinate\ keyword>^{+)}\}$,

where node1 is subordinate alert, node2 is principle alert,and +) *means*$\geq$1.

The algorithm Gen2-AASN shown in Figure 4.2 is used to construct 2-AASN from two atom alerts. It works under the Principal-subordinate relation. When one alert is in the subordinate phase, if its subordinate keywords are in a specific relationship with the principle alert, these two alerts are correlated. The process that the subordinate alert is replaced with the subordinate keyword is called substitution. For example, consider Snort alerts: *FINGER 0 query* and *FINGER redirection attempt*. The PCTCG format stream of these two alerts are shown in Figure 4.3.

Algorithm Gen2-AASN(alert1, alert2, SensorName);
Input: two alert messages and the IDS sensor name;
Output: 2-AASN of input alerts if a solution can be found.

1. node1 := alert1;
2. node2 := alert2;
3. $PCTCG_{alert1}$:= alert1 PCTCG format;
4. $PCTCG_{alert2}$:= alert2 PCTCG format;
5. alert1 phase := *subordinate* phase; node1::child := *keyword;*
6. alert2 phase := *principle* phase; node2::child := *case filler;*
7. let *MaxCasefiller* = total number of *case fillers;*
8. let *MaxKeyword* = total number of *keywords;*
9. **for** *i*:=1 to *MaxCasefiller* **do**
10. **for** *j*:=1 to *MaxKeyword* **do**
11. **if** semantic matching (::$keyword_i$ ::casefiller$_j$);
12. fill slot: *case filler* <*semantic role, keyword*>;
13. **end if**
14. **end for** (step 10)
15. **end for** (step9)
16. alert1 phase := *principle* phase; node1::child := *case filler;*
17. alert2 phase := *subordinate* phase; node2::child:= *keyword;*
18. repeat (9-15)
19. **return** (2-AASN);

Figure 4.2 Gen2-AASN algorithm.

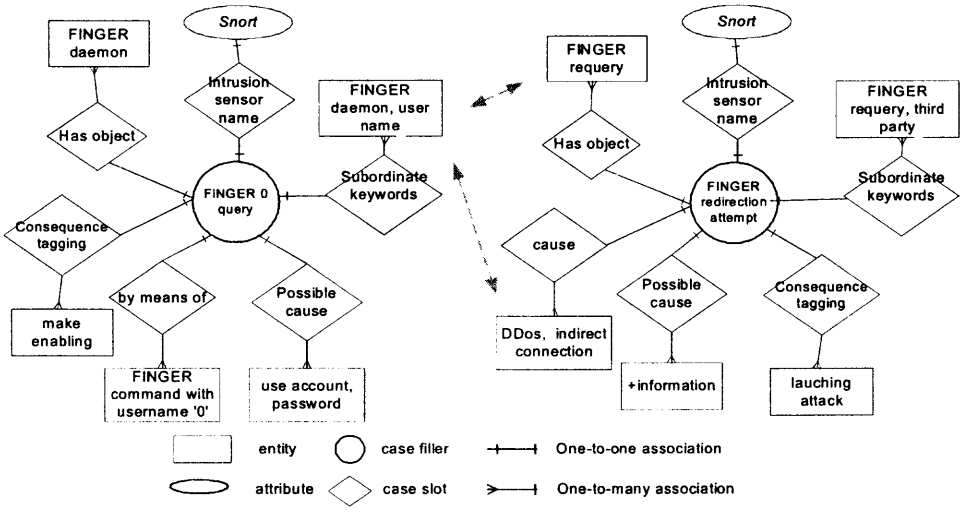


Figure 4.3 Semantic matching between PCTCG alert formats.

The edges of 2-AASN represents PCTCG semantic attribute or the label subordinate, and the nodes represent two atom alerts or their child nodes: *case filler* or the *subordinate keyword*. The format of 2-AASN is based on the 2-tuple slot: $< semantic\ attributes, case\ filler >$, or $< subordinate, subordinate\ keyword >$. The algorithm Gen2-AASN [4] was used to construct 2-AASN from two alerts. If there exists semantic attribute matching between the *case filler* and *subordinate keyword*, 2-AASN fills the slots: $node1 :: case\ filler < semantic\ attribute, node2 :: subordinate\ keyword >$ or $node2 :: case\ filler < semantic\ attribute, node1 :: subordinate\ keyword >$, and an arc between the *case filler* and *subordinate keyword* is generated. Using this procedure, the 2-AASN format of the Snort alerts is

$SN[node1, node2]=\{$

$node1:<subordinate, node1::username>,$

$node1:<subordinate, node1::FINGER\ daemon>,$

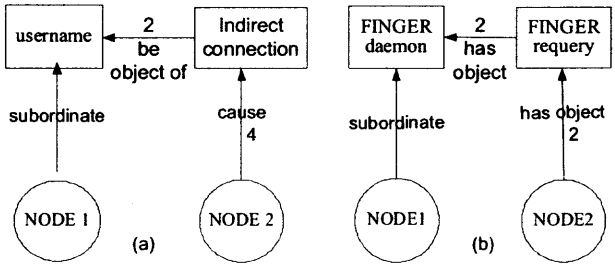
$node2:<cause, indirect\ connection>,$

$node2<has\ object, FINGER\ query>,$

$node2::indirect\ connection<be\ object\ of, node1::username>,$

$node2::FINGER\ query<has\ object, node1::FINGER\ daemon>.$

FINGER 0 query and *FINGER redirection attempt* is also represented by the semantic weighed network graph shown in Figure 4.4.



Node 1: FINGER 0 query
Node 2: FINGER redirection attempt

Figure 4.4 An example of 2-AASN.

4.1.2 Resolution Tree

In order to extract the attack scenario from 2-AASN, the resolution tree in Section 3.5, which works on CNF, was proposed. Initially, there are no connections between the case filler variables. When performing resolution, the variables are renamed so that they have no variables in common. Afterwards, the semantic roles R_1 and R_2 try to be fused into R_3. Here, $\star$ is the semantic role operator. Figure 4.5 illustrates an example of the resolution tree for alerts *FINGER 0 query* and *FINGER redirection attempt*. From $CC(Alert_2, Indirect\ conn)$ and $OB(indirect\ connection, username)$, the resolvent $EE(Alert_2, username)$ can be derived. (Note that the two semantic operations are fused into one.) Moreover, from $EE(Alert_2,$ $username)$ and $\Theta = username/Alert_1$, $EE(Alert_2, Alert_1)$ can be obtained, and $\neg EE$ $(Alert_2, Alert_1)$ and $EE(Alert_2, Alert_1)$ can be resolved into the empty clause.

Table 4.1 Semantic Role Fusion Operations

X	$X \star C = C$	$X \star C = X$	$X \star C = EE$	$X \star C = EB$
OH	OH,LH,LB,MH,MB,PB,CB	WM,WH	PC,CC	$\emptyset$
OB	OB,LH,LB,MH,MB,PC,CC	$\emptyset$	$\emptyset$	PB,CB
LH	OH,LH,LB,MH,MB,WM,WH	MB,WM,WH	PC,CC	$\emptyset$
LB	LB,MH	MH,WM,WH	PC,CC	PB,CB
MH	LH,LB,MH,PC,CC	LH,LB,WM,WH	$\emptyset$	PB,BB
MB	OB,LH,LB,MB,PB,BB	LH,LB,WM,WH	PC,CC	$\emptyset$
PC	PC,CC	OH,LH,MH,WM,WH	OB,LB,MB	$\emptyset$
PB	PB,BB	OH,OB,LB,PB,CB, WM,WH	$\emptyset$	LH,MH,MB
WM/WH	OH,OB,LH,LB,MH,MB,PC, CC,PB,CB,WM,WH	$\emptyset$	$\emptyset$	$\emptyset$
where OH: has object, OB: be object of, LH: has location, LB: be location of, MH: has instrument, MB: by means of, PC: (possible) cause, PB: be (possible) caused of, CC: cause, CB: be caused of, WM: meronymy, WH: Holonymy, EE:. Enable, EB: be enabled by.				

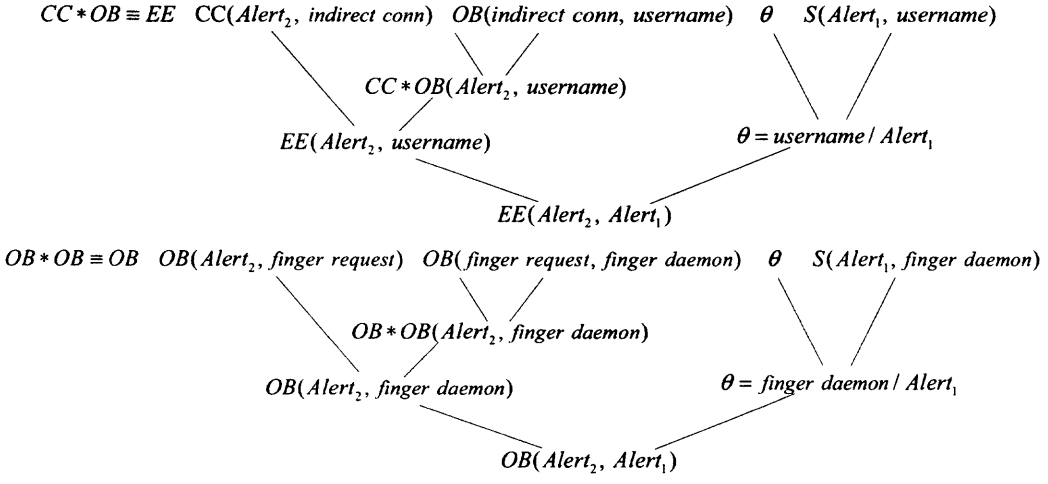


Figure 4.5 Example of resolution tree.

4.2 Correlation Rules

In the resolution tree, the correlation rules are defined and applied to extract the alerts correlations from 2-AASN. The semantic operator $\star$ is defined: *principle alert* $:<$ *semantic attribute*, *principle alert* :: *case filler* $>\star$ *principle alert* :: *case filler* $:<$ *semantic attribute*, *subordinate alert* :: *subordinate keyword* $>$. Table 4.1 shows the semantic role fusion operations. Some semantic roles cannot be fused and they are marked by $\emptyset$.

Table 4.2 Correlation Rules

Correlation rule	Match Phrase	Match process
(Possible) cause	Primary $\Rightarrow$ Secondary	A::filler matches (*possible*) *cause* to B::keyword
	Primary $\Leftarrow$ Secondary	B::filler matches (*possible*) *cause* to A::keyword
Enable	Primary $\Rightarrow$ Secondary	A::filler matches *enable* to B::keyword
	Primary $\Leftarrow$ Secondary	B::filler matches *be enaledby* to A::keyword
Instrument	Primary $\Rightarrow$ Secondary	A::filler matches *has instrument* to B::keyword
	Primary $\Leftarrow$ Secondary	B::filler matches *by means of* to A::keyword
Object	Primary $\Rightarrow$ Secondary	A::filler matches *has object* to B::keyword
	Primary $\Leftarrow$ Secondary	B::filler matches *be object of* to A::keyword
Part-whole	Primary $\Rightarrow$ Secondary	A::filler matches *meronymy* to B::keyword
	Primary $\Leftarrow$ Secondary	B::filler matches *holonymy* to A::keyword
Spatial	Primary $\Rightarrow$ Secondary	A::filler matches *has location* to B::keyword
	Primary $\Leftarrow$ Secondary	B::filler matches *be location of* to A::keyword

Consider the two parent nodes in 2-AASN: node A and node B. The case filler and subordinate keyword of node A and B are denoted as A :: *case filler*, A :: *keyword*, B :: *case filler* and B :: *keyword*, respectively. The (possible) cause, enable, instrument, object, part-whole, and spatial rules are defined. The enable rule takes place when one entity facilitates the other's attack process. The spatial rule describes the situation where one entity is surrounded by another entity but is not part of that entity. The (possible) cause, enable, instrument, and object rules are concerned with attack action "time" domain whereas the part-whole and spatial rules are related to "space" domain. Every correlation rule includes two matching phrases: the active way (primary $\Rightarrow$ secondary) and the passive way (primary $\Leftarrow$ secondary). Table 4.2 shows these correlation rules.

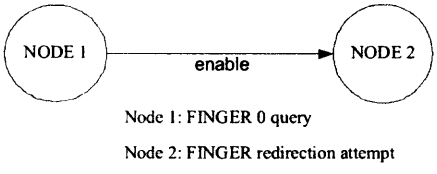


Figure 4.6 Semantic relation.

When extracting the correlations, if the sum of the weights of the principle alert's semantic attribute and the filled slot is greater than the semantic weight threshold (set to be 5), the $\star$ operation is performed. For example, the correlation between the two alerts, *FINGER 0 query* and *FINGER redirection attempt*, is shown in Figure 4.6. The attack scenario classes can be generated from the alert correlations. The attack scenario class is a directed graph where nodes are alert components, and arcs are semantic correlations. The similarity degree between attack scenario classes, S_1 and S_2, is defined as:

$$Sim_{1,2} = \frac{1}{2}(1 - \frac{\#\ of\ inserted\ or\ removed\ nodes}{\#\ of\ total\ nodes}) + \frac{1}{2}(1 - \frac{\sum_{i=1}^{n} w_i}{Max\{\sum_{j \in S_1} w_j, \sum_{k \in S_2} w_k\}}) \tag{4.1}$$

where w_i is the weight of the common links between S_1 and S_2, w_j is the link weight of S_1, and w_k is the link weight of S_2. For example, Figure 4.7 shows different Sim values with the changes of attack semantic dependency networks.

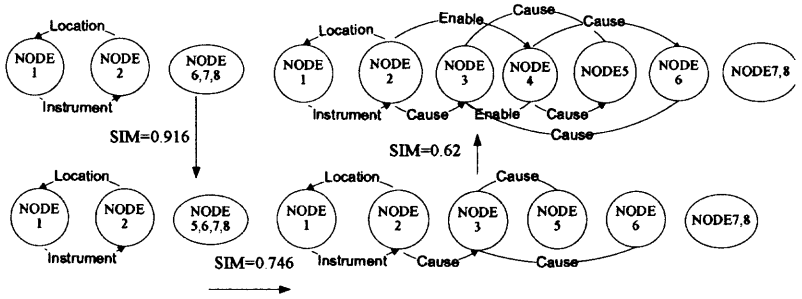


Figure 4.7 Attack scenario class similarity degree.

We also define the scenario stages, which are based on the sub-objective. The sub-objective is based on the consequence tagging class: gather information (try to gather the network information and application vulnerabilities), make enable (try to break into target and get control on the target), and launching attacks.

4.3 Alert Context Window

Since the attack scenario classes include all possible combinations of attack strategies and the attackers may only adopt a subset of the attack strategies to launch the attacks. To build the attack scenario, the alert contexts need to be considered. Because of the high volume of the alerts, it is not possible to consider correlation between the interested focus alert and all other alerts. Therefore, the alert context window size needs to be determined, and we only consider

the alerts within the context window and generate the attack scenarios from them. In NLP, context is used to determine the pronunciation, words collocation and words unambiguity [54, 55]. Here, the alerts context refers to the source and destination IP addresses, and timestamps within a certain context window. The alert context window (ACW) size is an important parameter of the alert context, which is the number of alerts before and after the interested focus alert. If the ACW size is too small, the correlated alerts would be absent. On the other hand, if the ACW size is too large, unnecessary computations and correlation noises (unrelated alerts) will be added. To extract attack scenarios, ACW should provide enough semantic information, and also restrains the correlation noises. However, there is no general method to define the size of the context window in natural language processing. In [56], the context window ± five can provide 95 % context for the linguistic collocations. Reference [57] also sets the window size to five to show the constraints between verbs and arguments. However, a small window size can identify the fixed expressions and word collocations that hold over a short range. Because of the interest in the semantic correlation between the alerts, a larger alert window size which can cover the semantic knowledge is preferable. The mutual information method [56] is used to determine the ACW size. Mutual information, which is a measurement of the associative strength between a pair of events, is defined to be:

$$MI(A, C, d) = \sum_{a \in A} \sum_{c \in C} p(a, c, d) I(a, c, d) \tag{4.2}$$

$$I(a, c, d) = log_2 \frac{p(a, c, d)}{p(a)p(c)} \tag{4.3}$$

where $a \neq c$ and $I(a, c, d)$ is the association ratio of two alerts a and c, and $p(a)$ and $p(c)$ are the probabilities of a and c, and $p(a, c, d)$ is the probability that a occurs before or after c at the distance d. If there is an association between a and c, $I(a, c, d) \gg 0$.

From Figure 4.8, it is clear that as the alert context window size increases, the degree of the mutual information decreases. At some distances, the associations are very small and do not decrease significantly, at which there are almost no associations between them. Within the ACW context range, the alerts and their semantic attributes build up the attack scenarios. To determine ACW, the alerts' mutual information along with the similarity degrees Sim are used. τ is set as the value of ACW where mutual information approaches 0.1. The mutual

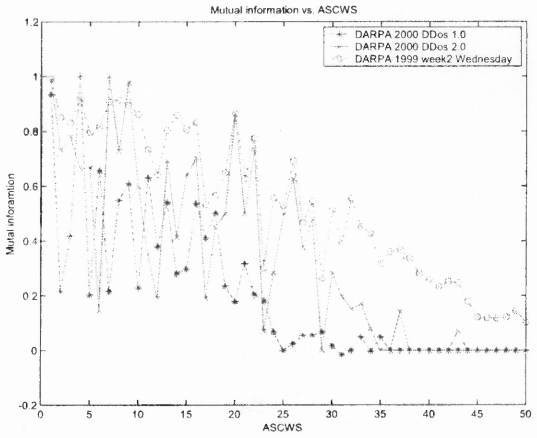


Figure 4.8 Mutual information for various ACW size.

information $MI(A, C, d)$ presents the alert frequency dependency between the alerts, while the attack scenario similarity degree Sim shows the changes of attack semantic correlations on the consecutive contexts. Therefore, the value of ACW is defined as follows:

$$ACW' = ACW - \lceil \alpha | ACW - \tau | - \beta Sim \rceil \tag{4.4}$$

where α and β are adjustable factors. In the simulations, α is set as 0.5 and β as 4, and initial ACW value is set as 35, 45, 60 for the three datasets shown in Figure 4.9, respectively, according to their diverse alert contexts. In the simulations, IDS sensors Snort [6] and Realsecure [58] were used, and datasets from [50] were tested on a PC (Pentium 4 processor at 2.8 GHz with 256M DDR memory). The gaol is to test:

1. Whether the time for PCTCG to convert the raw alerts into frame streams, and the FOL reasoning time can facilitate FAR-FAR real-time processing;
2. The performance of the attack scenario extraction.

FAR-FAR converts the alerts into IDMEF format and then into frame streams based on the PCTCG rules. The Snort Intrusion Event Database [59] includes 577 events. 550 snort alerts were selected from different class types to build up 550 PCTCG rules, and 117 Realsecure alerts were chosen from [58]. For Snort, five datasets (DDoS 1.0, DDoS 2.0, DARPA 1999 week 2 Monday, Friday, and 1998 week 6 Wednesday) were tested for the PCTCG converting time according to the alert number and the rule number. Figure 4.10(a) shows a plot of the PCTCG converting time versus the alert number for 35 PCTCG rules.

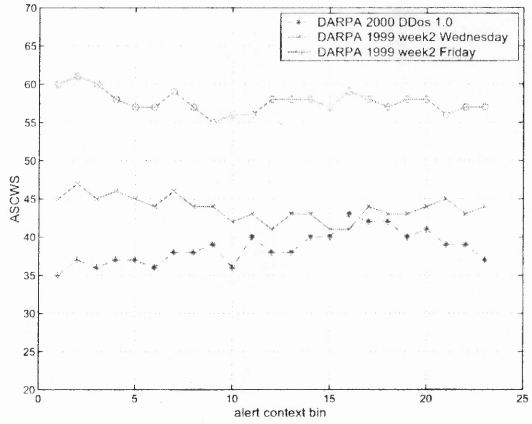
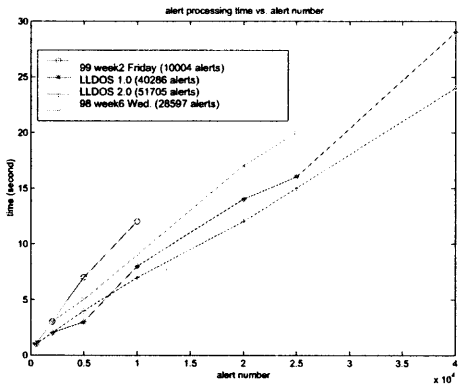


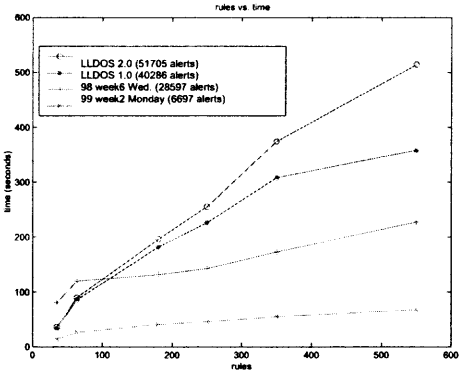
Figure 4.9 ACW vs. alert context bin.

Datasets DARPA 1999 week 2 Friday, and 1998 week 6 Wednesday spent more time than DDoS 1.0, DDoS 2.0 because they had relatively longer alert messages. The alerts were aggregated according to the IP address and consecutive time slot; these correlated alerts were all less than 2000. Afterwards, the PCTCG rule number was increased up to 550; the PCTCG processing time of the dataset with relative fewer alerts does not climb sharply with the increased PCTCG rule number, as shown in Figure 4.10(b). After aggregation, as shown in Figure 4.10(c), the processing time had sharply decreased to no more than 12 seconds, indicating that PCTCG conversion can be processed in real time. Figure 4.10(f) shows the alert reasoning time and the alert inter-arrival time. It is clear that the alert reasoning time is far less than the inter-arrival time. However, there are a few alerts whose reasoning time is larger than the inter-arrival time. This can be attributed to: (1) some attack actions cause more than one alerts (e.g., alerts *FINGER 0 query* and *FINGER requery* are all caused by *FINGER* action); (2) port scanning causes a number of alerts, making those alerts' inter-arrival time extremely small. However, the alerts by port scanning normally cannot be reasoned because they are usually uncorrelated with each other. Therefore, the FAR-FAR system can efficiently automatically reason the attack plans.

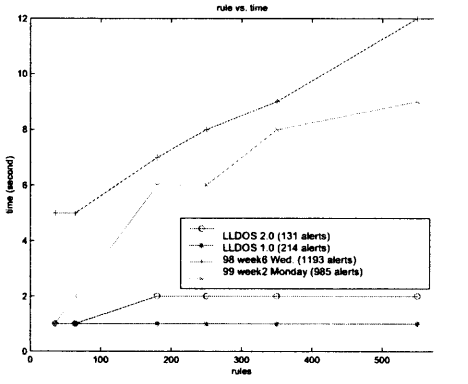
To test FAR-FAR's reasoning performance, DARPA 1999 week 2 Wednesday, DDos 1.0, and DDos 2.0 datasets were tested. In Figure 4.10(d), at some ACW distances, the normalized mutual information are very small and do not decrease significantly, and the initial ACW values for those datasets were set as 35, 45, 60. Figure 4.10(e) shows the real-time adjustment according to their diverse alert contexts.



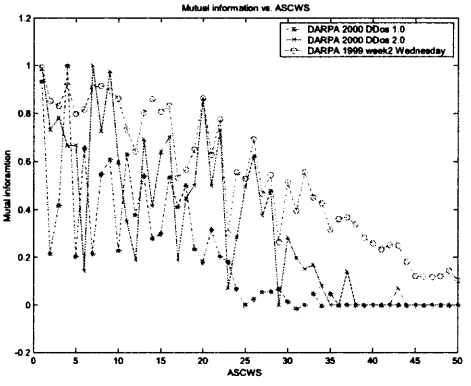
(a) PCTCG Processing Time vs. Alerts



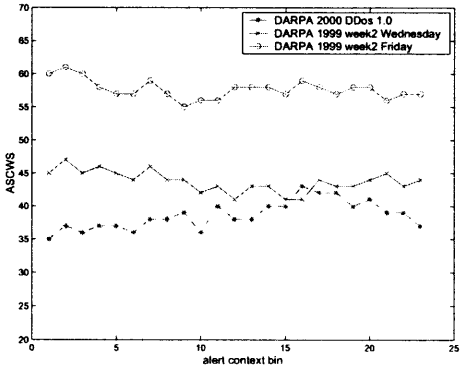
(b) PCTCG Processing Time vs. Rules



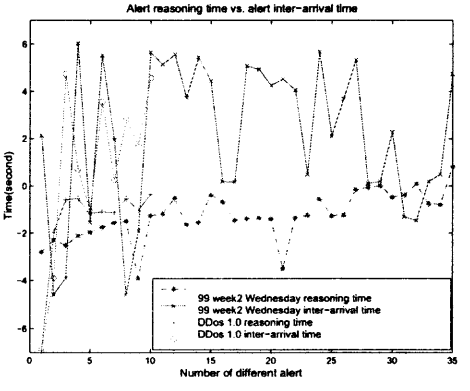
(c) Reasoning vs. Alert Inter-arrival Time



(d) Mutual Information vs. ACW



(e) ACW vs. Alert Context Bin



(f) Reasoning vs. Alert Inter-arrival Time

Figure 4.10 FAR-FAR simulation results.

4.4 Alert Semantic Vector

In NLP, text categorization means the assignment of free text documents to one or more predefined categories based on their contents. A number of statistical classification and machine learning techniques have been applied to text categorization [60, 61]. In this section, mutual information is used to determine the alert semantic context range.

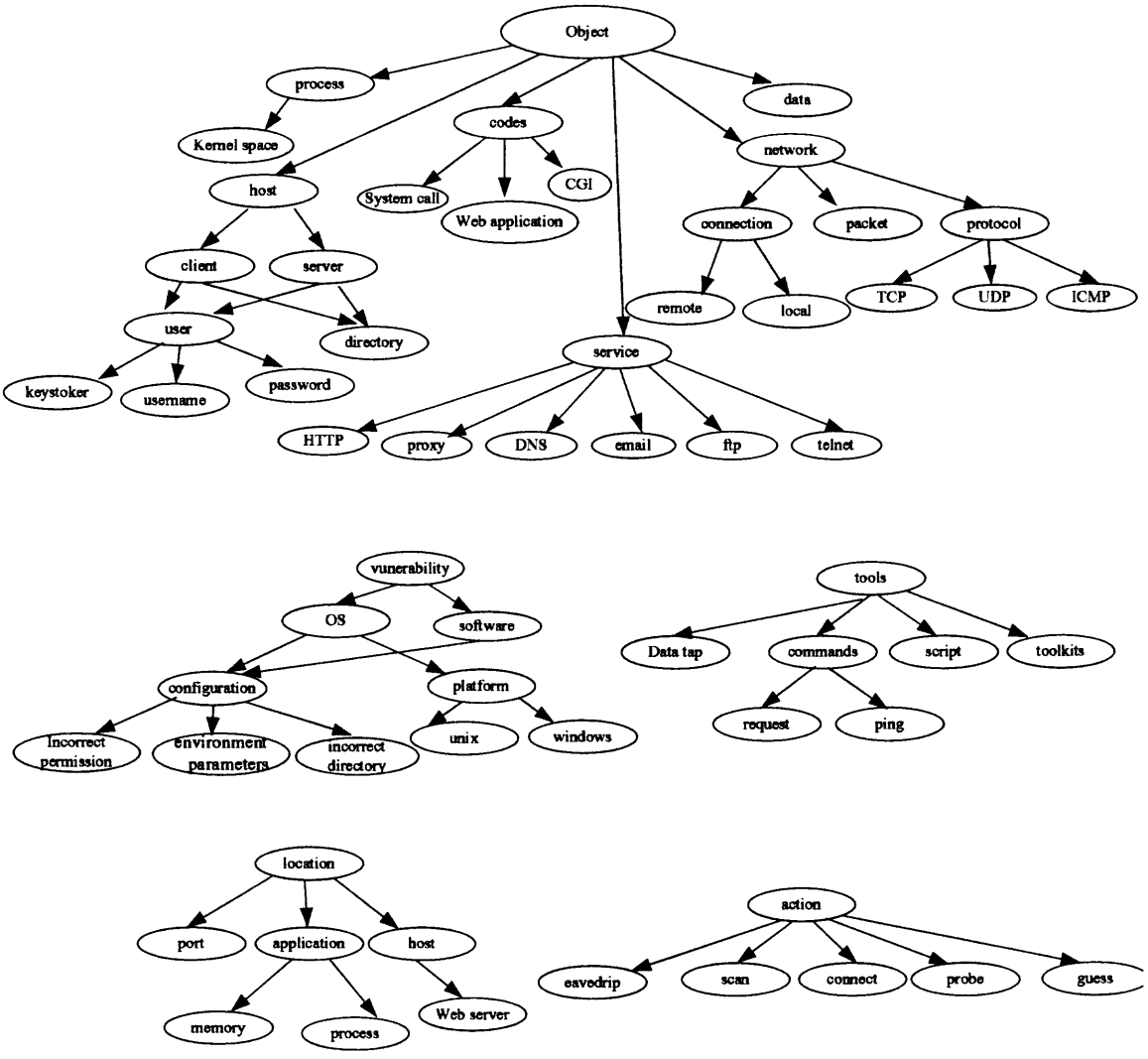


Figure 4.11 Abstraction of the "gain information" ontology.

Based on the attack ontology and alert contexts, alerts are represented as attack semantic space vectors. A text categorization technique is then applied to categorize the intrusion stages. The retrieval method is based on measuring the similarity between queries and documents.

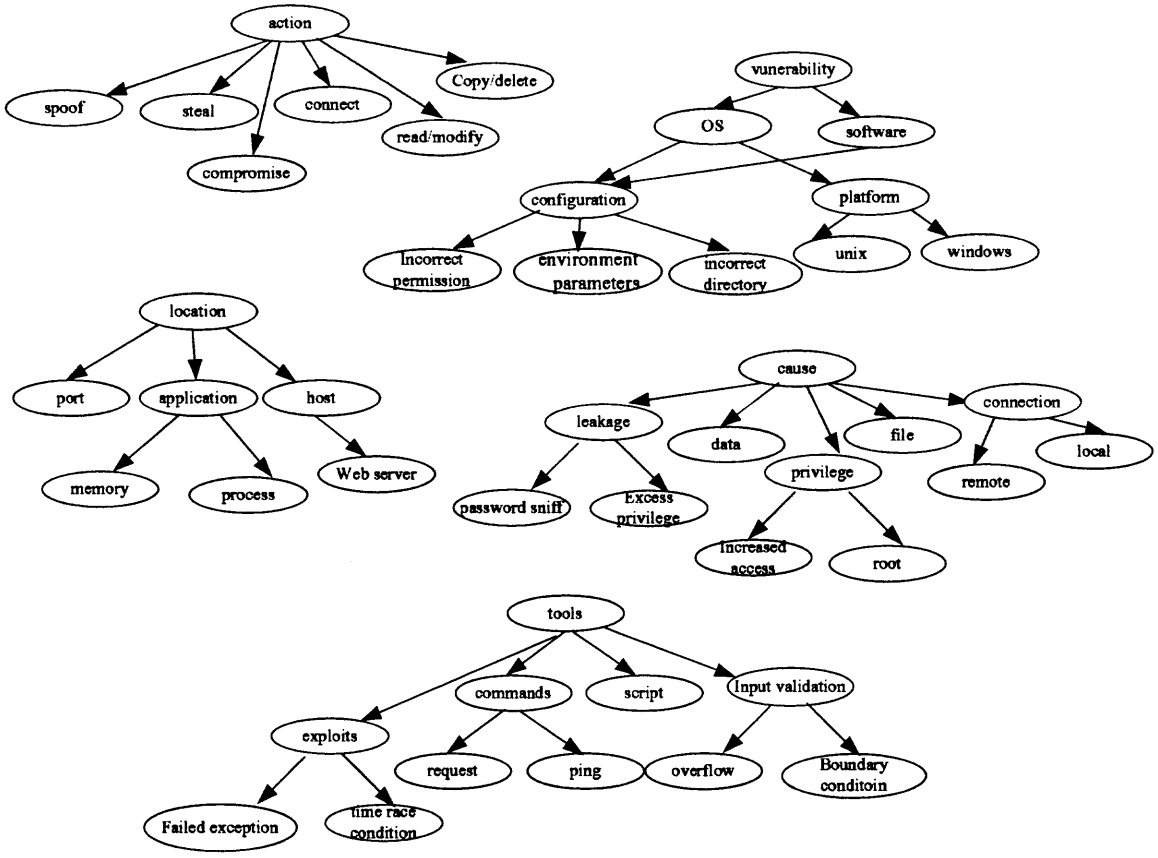


Figure 4.12 Abstraction of the "control" ontology.

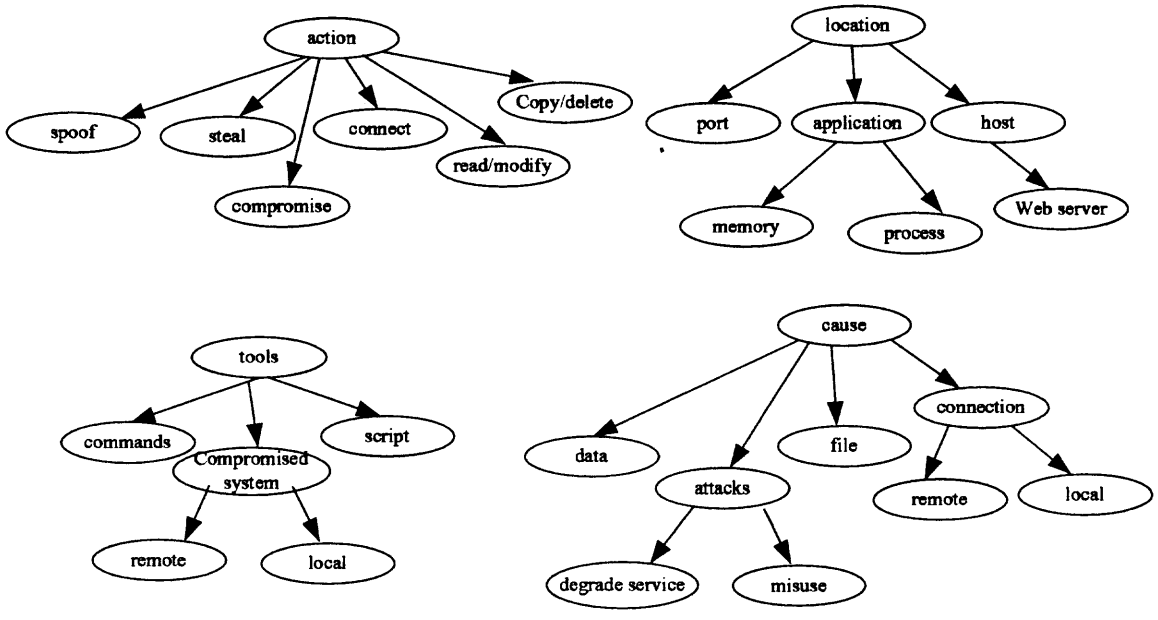


Figure 4.13 Abstraction of the "launching attacks" ontology.

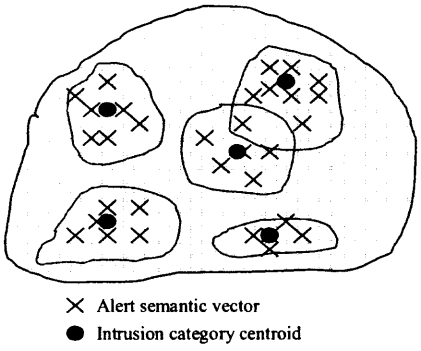


Figure 4.14 Vector space model.

The vector space model method in [60] is used for the alert categorization. First, alerts are divided into segments by consecutive timeslots, and each segment is viewed as a "document". The alerts in each document is represented by the PCTCG frame subordinate keywords, a features defined by the attack ontology. Figures 4.11, 4.12, and 4.13 show the fragments of the FAR-FAR attack ontology, which is the extension from the previous research works in [35, 36, 37]. Afterwards, the feature term frequency (tf) in each document is counted, and a feature weight is calculated using the tf and the inverse document frequency (idf) method (TF-IDF) [62]. The raw tf of a feature term in a alert segment is multiplied by the term's idf weight:

$$w_{kd} = tf_{kd} \cdot idf_k = tf_{kd} \cdot \log\left(\frac{N}{D_k}\right), \tag{4.5}$$

where tf_{kd} is the frequency with which feature $k, 1 \leq k \leq n$, occurs in the alert segment d, N is the total number of alert segments in the log corpus, and D_k is the number of segments containing feature k. Afterwards, the vector is normalized:

$$W_k = \frac{w_{kd}}{\sqrt{\sum_{k=1}^{n} w_{kd}^2}} \tag{4.6}$$

Finally, the similarity between an alert's vector and an intrusion category's centroid is measured, as shown in Figure 4.14. The intrusion category's centroid is calculated from the training data. Three intrusion categories are defined: *gain information* ($\mathcal{G}$), *making enable* ($\mathcal{M}$), and *launching attacks* ($\mathcal{L}$).

In Figure 4.15, the feature's tf values are stored in the tf hash table. Suppose there exist three intrusion stages, each of which includes several sub-stages. Every sub-stage has a

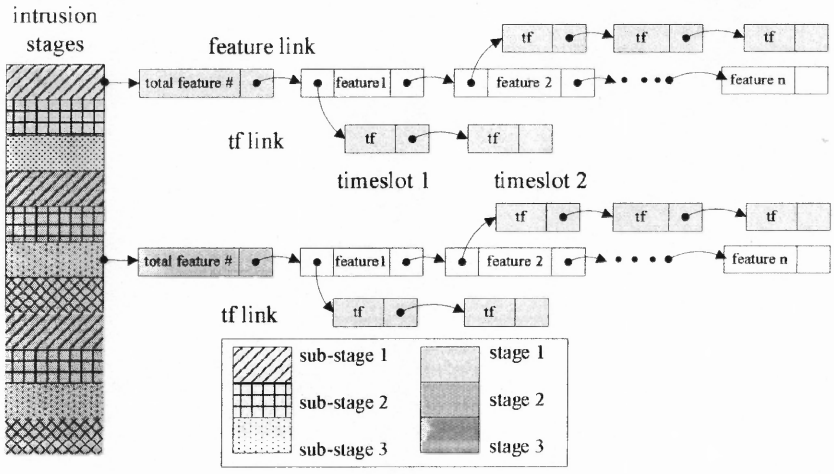


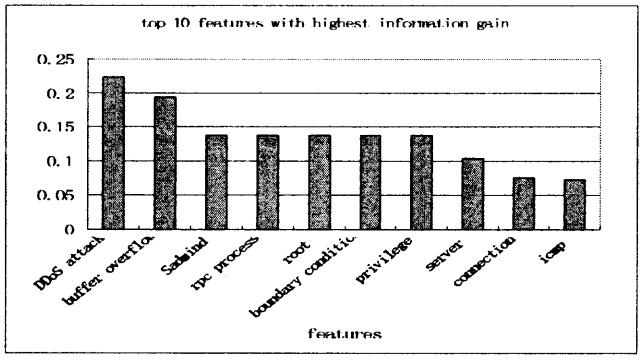
Figure 4.15 Feature tf hash table of intrusion stages.

feature link, connected by a set of tf links for every feature. From the tf hash table, tf and idf values can be calculated (D_k is the total number of link nodes for feature k). Cosine-based similarity between two n dimensional semantic space vectors, an alert a's semantic vector $\vec{i}$, and a category vector $\vec{j}$, is measured by:

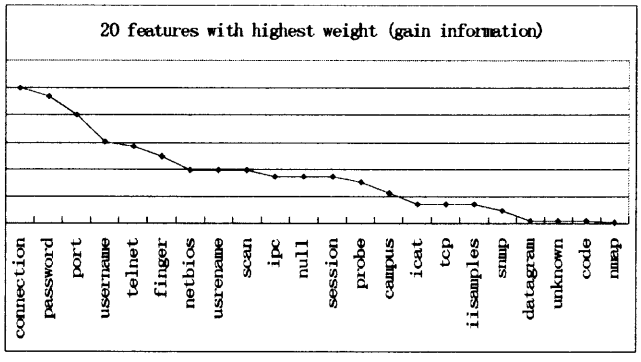
$$sim(\vec{i}, \vec{j}) = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\| \vec{i} \|_2 * \| \vec{j} \|_2} \tag{4.7}$$

where a belongs to the category with the highest value.

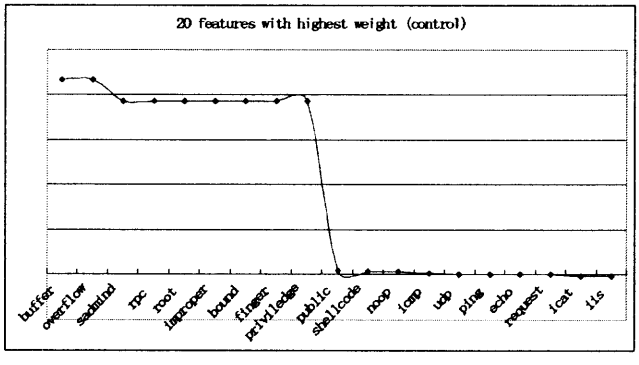
To evaluate the intrusion categorization performance, three classification methods (TF-IDF, Naive Bayes, and KNN) [63] were compared using the classification tool. The DARPA datasets were mixed. 60% of the DARPA datasets were used as training data, and the remaining 40% of the data were tested. The categorization results showed that TF-IDF had 67.6% for intrusion stage $\mathcal{G}$, 90.1% for $\mathcal{M}$, and 94.5% for $\mathcal{L}$. That is, TF-IDF has better performance than Naive Bayes and KNN methods. The reason that TF-IDF has higher classification ratio in $\mathcal{M}$ and $\mathcal{L}$ stages is that the features describing $\mathcal{M}$ and $\mathcal{L}$ activities are more specific than $\mathcal{G}$. Figure 4.16 shows the weight ranking of those features.



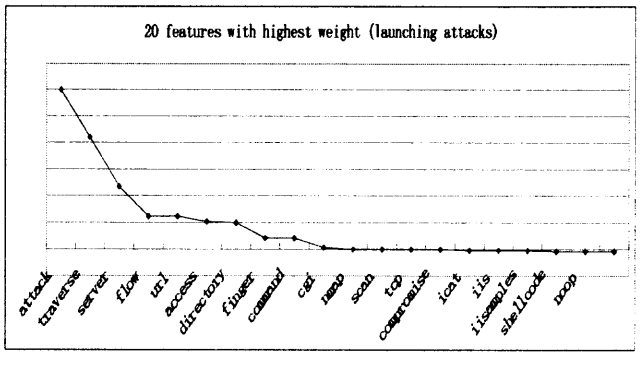
(a) Top 10 features with highest information gain.



(b) 20 features with highest weights in stage 1.



(c) 20 features with highest weights in stage 2.



(d) 20 features with highest weights in stage 3.

Figure 4.16 FAR-FAR simulation results.

CHAPTER 5

SPREADING ACTIVATION AND SEMANTIC QUERY

The current IDS monitoring system can hardly provides precise answers for the attack scenario related queries. In [64, 65], the conjunctive query approach for Semantic Web was presented and will be the basis in defining the semantic attack knowledge query language. The following are the definitions of the semantic queries.

Definition 5.0.1 (Valid query term for $\mathcal{AKB}$) *A term $\mathcal{Q}$ with the form: $a : C$ or $< a, b >: R$ where C is the set of nodes in the attack scenario, and R is the set of correlation of attack scenario, is called a valid query term for $\mathcal{AKB}$ if it is satisfied by Attack Interpretation $\mathcal{AI}$, denoted as $\mathcal{Q} \uplus \mathcal{AKB}$.*

Definition 5.0.2 (Valid conjunctive query for $\mathcal{AKB}$) *A conjunctive query statement $CQ = Q_1 \wedge \cdots \wedge Q_n$, where Q_i , $1 \leq i \leq n$ is query term, is called a valid conjunctive query for $\mathcal{AKB}$ if $Q_i \uplus \mathcal{AKB}$, denoted as $\mathcal{CQ} \uplus \mathcal{AKB}$.*

Definition 5.0.3 (Various conjunctive query statements) *Four types of statements are defined as follows:*

- *Does alert node x which belongs to the set of nodes in the attack scenario?* $\rightarrow x : FC$
- *What is the associated alert node y of alert node x for the semantic relation R?* $\rightarrow \vartheta_{\exists R.C}(x)$ *where* $\vartheta_{\exists R.C}(x) = \exists y.R(x, y) \wedge C(x)$
- *Do correlation between x and y belongs to R?* $\rightarrow < x, y >: R$
- *Derive the attack paths from the initial node x to the destination node y?* $\rightarrow < x, z_1 >: R_1 \wedge < z_1, z_2 >: R_2 \wedge \cdots < z_n, y >: R_{n+1}$

5.1 Spreading Activation

In the query model, the attack semantic query is used to enable the administrator to query the intrusion states of the network. The semantic relationships can be queried and discovered through traversing sequence of links among the entities of interests. Since the attack scenario classes include all possible combinations of the attack actions, the attack scenario instances are generated based on the alert context which describes the specific attack scenario. Using the weight mapping technique, a weight is assigned to each relation instance to express the associated strength between two nodes. In [65], the cluster weight mapping was introduced and the formula used to calculate the weight is:

$$W(C_j, C_k) = \frac{\sum_{i=1}^{n} n_{ijk}}{\sum_{i=1}^{n} n_{ij}} \tag{5.1}$$

The value n_{ij} denotes the concept C_j is related to C_i. The value n_{ijk} denotes that both concepts C_j and C_k are related to C_i.

The Spread Activation (SA) technique [65] is used in the semantic query model for attack scenario knowledge retrieval. SA searches for the paths connecting the start nodes and the destination nodes based on an evaluation criterion. For example, given the initial set of nodes and their activation values, the activation flows through the network reaching other concepts which are closely related to the initial concepts. If the current node passes certain constraints and not all its neighbors are activated, it propagates its activation value to its neighbors. The activation strength decreases in proportional to the distance in between. The decay factor is defined to reduce the activation strength within the propagation process. The activation input into a node can be represented by the following formula:

$$I_j = \sum_{i=1}^{m} O_i W_{ij}(1-a) \tag{5.2}$$

where I_j is the total input of node j, O_i is the output of node i connecting to node j, a is the decay factor, and W_{ij} is the weight associated to the link connecting node i to node j by the weight mapping. The output activation of node O_i is determined by:

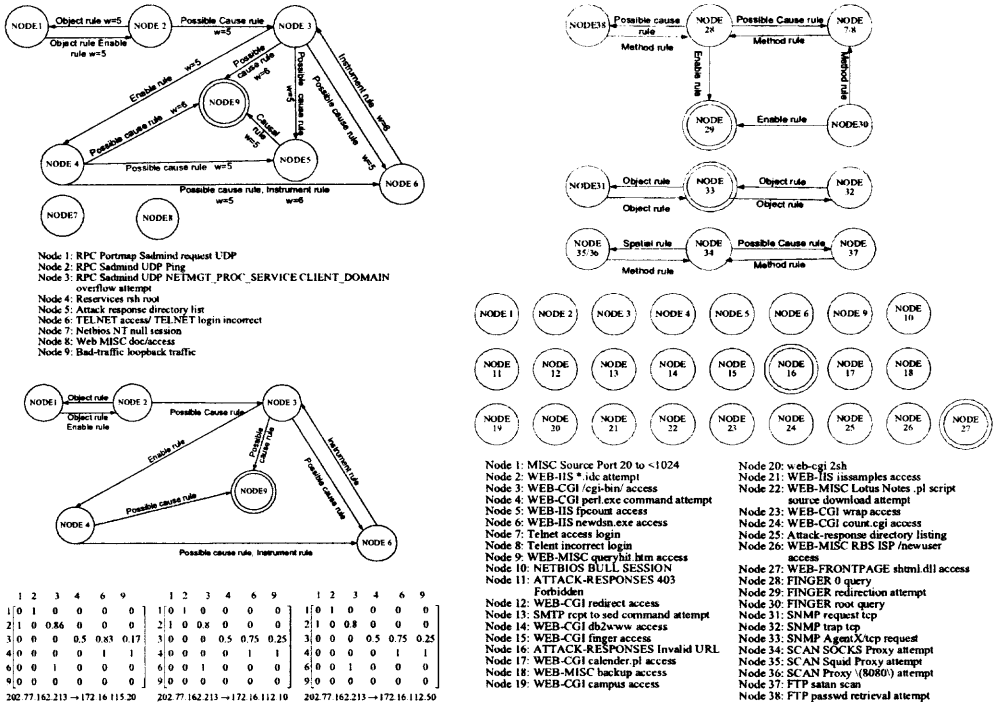
$$O_i = f_i(I_i) \; where \; f_i(I_i) = \begin{cases} I_i & I_i > T \\ 0 & I_i \leq T \end{cases} \tag{5.3}$$

where T is the threshold. The output value is fired to all nodes connected to the active node. The spreading phase of the pulse consists of the flow of activation waves from one node to all other nodes connected to it. This cycle goes on until the termination condition is met. The end result of the SA process is the activation level of each node in the network at the termination time.

For IDS, at the semantic query interface, the user can express the attack scenario knowledge query in terms of the attack phrases. With the help of attack phrase synonym knowledge base, the query model searches for all the nodes in the attack instance network whose *subordinate keywords* match the attack phrases or the phrases' synonyms. Those matched nodes are supplied to SA as the initial nodes and their initial activation values

are set to 1. The user can also define the terminal states (the default terminal states are the attack launching nodes in the scenarios) to stop the SA process. The set of nodes obtained at the end of the propagation are presented to the user as the result of the semantic search.

5.2 Semantic Query and Simulations



(a) Attack Scenario of LLDOS 1.0. (b) Attack Scenario Class of 99 week 2 Wed.

Figure 5.1 Simulations of DARPA LLDOS 1.0 and 99 week 2 Wednesday datasets.

The datasets in the simulations are LLDOS 1.0 and 1999 week 2 Wednesday. For LLDOS 1.0 dataset, two IDS sensors: Snort and RealSecure are used and the generated alerts are fused. First, the tcpdump dataset is replayed and aggregated according to the source IP address, target IP address, and the consecutive time slot. Afterwards, 2-AASN of the alerts is built up, and the correlation between them is extracted by the semantic attribute operation to form the attack scenario class (the semantic match weight threshold is set to 5). The simulation results showed that there were three attack scenario instances in LLDOS (attacker 202.77.162.213 → victim 172.16.115.20, 202.77.162.213 → victim 172.16.112.10, and 202.77.162.213 → victim 172.16.112.50). As shown in Table 5.1, FAR-FAR can decrease the false alarm sharply. For example, after aggregation, the alert number decreased to 0.32%, and there are 0.13% alerts in the attack scenario instance. Furthermore, 0.042% aggregated alerts

were in the *gather information* attack stage, 0.087% aggregated alerts in the *making enable* stage, and 0.0074% aggregated alerts in the *launching attack* stage. The attack scenarios of two datasets are shown in Figure 5.1. The scenario class of LLDOS 1.0 includes six focus alerts (node 1, node 2, node 3, node 4, node 6 and node 9), and attack instance weight matrices are also presented in Figure 5.1(a). The attack scenario class of 1999 week 2 Wednesday dataset is shown in Figure 5.1(b).

Table 5.1 Simulation Results of Alerts Number in Two Alert Datasets (w=5)

Data set	Snort alert	Aggregated alert	Alerts in instance	Gather information	Make enable	Launch attack
LLDOS 1.0	40288	0.32%	0.13%	0.042%	0.087%	0.0074%
99 week2 Wed.	31601	8.38%	0.105%	0.035%	0.019%	0.051%

Table 5.2 Simulation Results of Evaluation Parameters in Two Alert Datasets

Parameters	LLDOS 1.0			99 week 2 Wednesday		
	w=4	w=5	w=6	w=4	w=5	w=6
attack class missing focus alert	0	0	0	2	2	2
attack class false focus alert	0	0	0	4	2	2
attack class missing attack links	0	0	1	2	2	2
attack class false attack links	6	3	2	7	5	4
node instance rate	0.86	0.86	1.00	0.69	0.82	0.75
link instance rate	0.44	0.77	1.00	0.50	0.56	0.50

Second, the datasets were simulated under different w values to evaluate the performance of SIM. w is set as four, five and six, respectively. The simulation results are shown in Table 5.2. For example, when $w = 5$, the *node instance ratio* is 0.86, and the *link instance ratio* is 0.50, implying that the generated attack scenario class can describe the actual attack plan in the LLDOS 1.0 dataset well without causing high false actions and false semantic relations. Moreover, there is no attack class which misses the focus alert, no attack class which produces false focus alerts, and no attack class which misses the attack step, implying that FAR-FAR can "denoise" the unrelated alerts without missing attack steps.

Third, for the attack semantic query, two queries were simulated on the LLDOS 1.0 dataset. Suppose the network administrator knows certain hosts have the vulnerability of sadmind service, and wants to know whether this vulnerability can be used to cause the DDoS attacks. Thus, in query 1, he/she inputs the attack state "admin", sets the DDoS as the terminate state, and submits this query to the semantic search model. In query 2, he/she wants to know what consequence the *RPC Sadmind overflow* event can produce.

Table 5.3 Semantic Search Results of Query 1 and Query 2 (w=5)

Query	Semantic Search Path	Node activation
Query 1	Initial set:① Terminate set: ②③④ ⑤⑥⑨ ①→②→③→⑥→③→④→⑨	202.77.162.213 → 172.16.115.20 : 1.0 → 0.9 → 0.69 → 0.52 → 1.16 → 0.52 → 0.47 202.77.162.213 → 172.16.112.10 : 1.0 → 0.9 → 0.65 → 0.44 → 1.05 → 0.46 → 0.43 202.77.162.213 → 172.16.112.50 : 1.0 → 0.9 → 0.65 → 0.44 → 1.05 → 0.46 → 0.43
Query 2	Initial set:③ Terminate set: ④⑥ ③→⑥, ③→④	202.77.162.213 → 172.16.115.20 : 1.0 → 0.75, 1.0 → 0.46 202.77.162.213 → 172.16.115.20 : 1.0 → 0.48 1.0 → 0.46 202.77.162.213 → 172.16.112.50 : 1.0 → 0.48 1.0 → 0.46

Table 6 shows the query results and its semantic search weights. For query 1, three attack scenario instances have identical terminal set, and the attack steps from, discovering sadmind vulnerability to launching attack (described in [50]) are as follows: Hosts running sadmind service are probed, by using the "ping" option of the sadmind exploit program (②); the attacker tries to break into these hosts by remote buffer-overflow attack (③); to test whether or not a break-in was successful, the attacker attempts several login commands via telnet (⑥); then, the attacker installs the ".rhosts" file(④), and finally launches the DDos attacks(⑨). The semantic search path in Table 5.3, and their activations can clearly inform the network administrator about the above attack plan. For query 2, the consequence of the *RPC Sadmind overflow* event is enabling the attack to install the ".rhosts" file by telnet(③→⑥, and ③→④).

5.3 Comparison with Related Works

In this section, related correlation works are compared with FAR-FAR. First, the performance of extracting attack scenario is evaluated. Two parameters are defined: the correct correlation ratio and the number of alerts in the attack scenario. Then the parameter of the false alarm rate is evaluated. The correct correlation ratio is defined as

$$correct\ correlation\ ratio = \frac{\#\ of\ correct\ correlated\ alerts}{total\ \#\ of\ correlated\ alerts} \tag{5.4}$$

We compared the performance of extracting attack scenario between FAR-FAR and the Apriori method [66]. Apriori is a method to extract the highest possible association rules. Let $I = \{i_1, i_2, \cdots, i_n\}$ be a set of items, and let D be a set of the transactions where each transaction T is a set of the items such that $T \subset I$. An association rule is defined as: $X \rightarrow Y$, where $X \in I, Y \in I$ and $X \cap Y = \emptyset$. The association rule $X \rightarrow Y$ has the confidence degree

$c\%$ if $c\%$ of transactions in D containing X also contain Y. The association rule $X \rightarrow Y$ has the support degree $s\%$ if $s\%$ of the transactions in D contain $X \cup Y$ [66]. Given a set of transactions D, the problem of mining association rules is to generate all the association rules that have s and c greater than the user-specified minimum support and minimum confidence degrees, respectively.

Table 5.4 Correct Correlation Ratio Between TIAA and FAR-FAR.

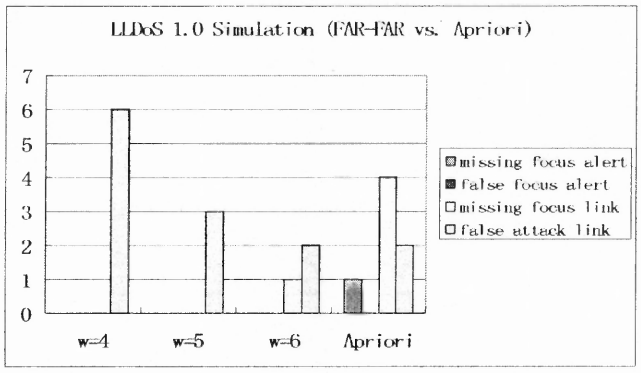
System	LLDoS 1.0	LLDoS 2.0	99 week2 Wednesday
TIAA	0.93	0.67	-
FAR-FAR	0.86	0.78	0.82

Table 5.5 False Alarm Rate Between TIAA and FAR-FAR.

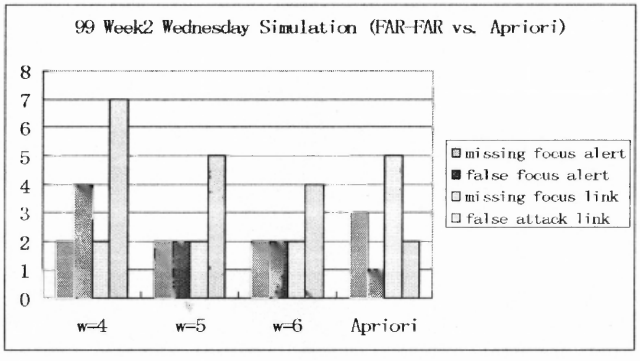
System	LLDoS 1.0	LLDoS 2.0	98 week2 Wed.	99 week2 Wed.
TIAA	5.26%	6.82%	-	-
FAR-FAR	0.31%	0.25%	4.17%	8.38%

However, in this dissertation, the Apriori method is not used for two reasons. First, for those duplicated alerts indicating the DDoS attacks, the aggregation process usually eliminates them causing very low support degree, which in terms causes those alerts missing in the attack scenario. Second, for the very common alert, such as "telnet" or "scan" alerts, since they can be associated with a number of alerts, the association rules containing them will have very low "confidence" degree, which leads to high "missing focus alerts" and "missing attack links". In Figure 5.2, we compared the performance of extracting the attack scenario between FAR-FAR and the Apriori method. It is clear that the Apriori method had much higher "missing focus alerts" and "missing attack links" value. For the false attack links, since Apriori extracted much less attack correlations than FAR-FAR, it produced a lower value.

As shown in Table 5.4, for the three datasets, FAR-FAR has a stable correct correlation ratio around 0.80, whereas the correlation toolkit TIAA in [20] has 0.93 and 0.67, respectively. One of the reasons that TIAA has higher correct correlation ratio than FAR-FAR is that TIAA has 44 correlated alerts whereas FAR-FAR has only 14 alerts. Figure 5.3 shows that these 14 alerts belong to three intrusion stages. In Table 5.5, the false alarm rate of FAR-FAR is much less than that of TIAA, implying that based on the semantic processing, FAR-FAR can



(a) LLDoS 1.0 Simulation (FAR-FAR vs. Apriori)



(b) 99 Week2 Wednesday Simulation (FAR-FAR vs. Apriori)

Figure 5.2 Simulation comparison between FAR-FAR and Apriori method.

AS (DARPA 2000) = {
objective name : *attack 172.16.115.20, 172.16.112.10, 172.16.115.50*

gather information
RPC Portmap Sadmind request UDP, *enable* *RPC Sadmind UDP Ping,*
< 202.77.162.213,172.16.115.20,10 : 08 : 07.354091 > *< 202.77.162.213,172.16.115.20,10 : 08 : 07.359636 >*
< 202.77.162.213,172.16.112.10,10 : 15 : 10.023115 > *< 202.77.162.213,172.16.112.10,10 : 15 : 10.026586 >*
< 202.77.162.213,172.16.115.50,10 : 15 : 10.098496 > *< 202.77.162.213,172.16.115.50,10 : 15 : 10.102257 >*

.....
RPC Sadmind UDP NETMGT_PROC_SERVICE CLIENT_DOMAIN overflow *cause* *Telnet access,*
< 202.77.162.213,172.16.115.20,10 : 33 : 10.621429 > *< 202.77.162.213,172.16.115.20,10 : 33 : 14.728748 >*
< 202.77.162.213,172.16.115.20,10 : 33 : 20.923039 > *< 202.77.162.213,172.16.115.20,10 : 33 : 23.011892 >*
< 202.77.162.213,172.16.115.20,10 : 33 : 27.165722 > *< 202.77.162.213,172.16.115.20,10 : 33 : 32.470221 >}*
.....
make enable
RPC Portmap Sadmind request UDP *cause* *RPC Sadmind UDP NETMGT_PROC_SERVICE CLIENT_DOMAIN overflow*
< 202.77.162.213,172.16.115.20,10 : 33 : 10.611612 > *< 202.77.162.213,172.16.115.20,10 : 33 : 10.621429 >*
< 202.77.162.213,172.16.115.20,10 : 33 : 12.642958 > *< 202.77.162.213,172.16.115.20,10 : 33 : 12.652687 >*
< 202.77.162.213,172.16.115.20,10 : 33 : 18.875888 > *< 202.77.162.213,172.16.115.20,10 : 33 : 18.885651 >*
< 202.77.162.213,172.16.115.20,10 : 33 : 20.913357 > *< 202.77.162.213,172.16.115.20,10 : 33 : 20.923039 >*
.....
Telnet access, *instrument* *RSERVICES rsh root,*
< 172.16.115.20,202.77.162.213,10 : 50 : 01.819752 > *< 172.16.115.20,202.77.162.213,10 : 50 : 04.146207 >*
< 172.16.112.10,202.77.162.213,10 : 50 : 21.064056 > *< 172.16.112.10,202.77.162.213,10 : 50 : 22.146207 >*
< 172.16.115.50,202.77.162.213,10 : 50 : 37.923074 > *< 172.16.115.20,202.77.162.213,10 : 50 : 38.176538 >*
.....
launching attacks
bad traffic loopback traffic
< 202.77.162.213,172.16.115.20,10 : 33 : 29.223090 >}

Figure 5.3 Attack description of LLDoS1.0 dataset.

efficiently "denoise" unimportant alerts. In [67], CAIDS was used to test the same datasets from [50], and its false alarm rate was around 5%. Therefore, FAR-FAR's reasoning capacity for the attack scenario can compete well with current correlation tools. Furthermore, based on the semantic attack ontology, FAR-FAR can also provide the functions of semantic reasoning and alert categorization.

CHAPTER 6

ANOMALY DETECTION BY HISTOGRAM FEATURE VECTOR

A number of recent studies of real traffic from modern networks demonstrated that real traffic exhibits statistical self-similarity and heavy-tail, and the traditional models such as Poisson or Markovian, are basically not applicable to model self-similar traffic. In the heavy tail distribution, most of the observations are small, but most of the contribution to the sample mean or the variance comes from the few large observations [68]. On the other hand, the dramatic expansion of networking applications makes network security a pressing issue. As increasingly more network facilities are connected to the internet, their vulnerabilities make it easy for an attacker to initiate attacks. For example, DDoS has caused a huge economic loss to the victims. Therefore, the detection of traffic anomaly is important to the security of modern networks. Since traffic anomalies do not have rigid rules, capturing them is fundamentally essential to enhance the robustness and survivability of communication networks.

In this chapter, the work statistically analyzes the histograms of ordinary-behaving bursty traffic traces, and apply the Multi-Time scaling Detection (MTD) to detect the bursty traffic anomalies. Given a traffic trace including some bins, HFV is a vector composed of the histogram frequency of every bin. Since the self-similar traffic traces follow the heavy tail distribution, their HFVs contain left skewness, resulting from the bulk of the values being small but with a few samples having large values. Research work in [69] showed the simply aggregating self-similar traffic traces will not decrease the heavy tail characteristic significantly. Therefore, the aggregated traffics on the edge network devices of high speed network will present self-similarity if the incoming traffics are bursty, and the high speed network traffic anomalies can be detected by large HFV anomaly deviation of self-similar reference traffic trace. Furthermore, since every traffic bin contains different burstiness characteristics, the burstiness compensation parameter γ is introduced to smooth the deviation error caused by those burstiness differences.

6.1 Self-similarity Traffic Model

Ever since Internet has been developed, both the transmission speed and the service on the wide-area internet remain a crucial problem. Recent works in traffic analysis have shown that traffic streams traversing modern networks are self-similar over several time scales from microseconds to minutes [68, 69, 70, 71, 72]. Self-Similarity describes the phenomenon where a certain property of an object is preserved with respect to scaling in space or in time. Scaling behaviors of the Internet traffic have a significant impact on network performance. For example, self-similar traffic causes worse network performance than the Short Range Dependence traffic does.

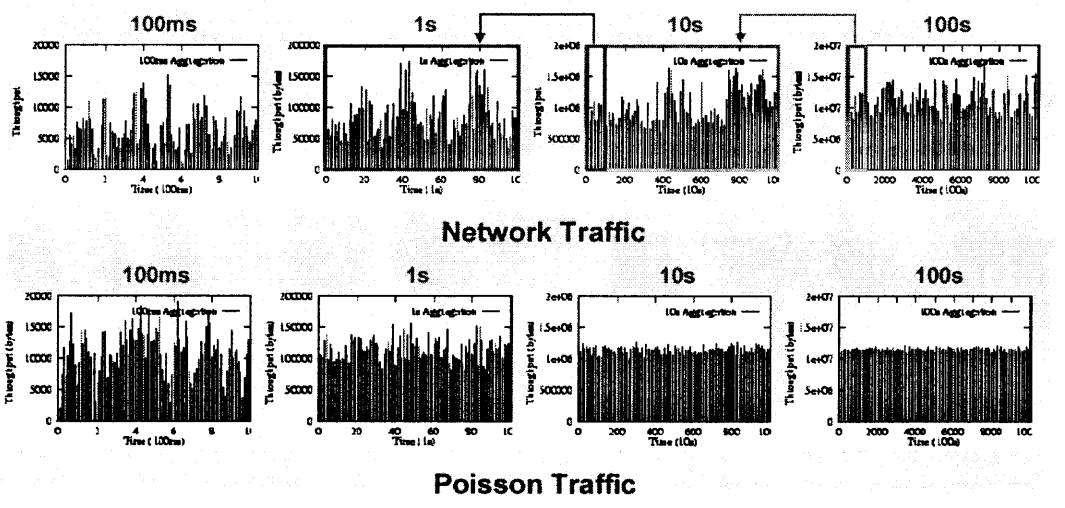


Figure 6.1 Self-similarity in traffic measurement.

In Figure 6.1, the self-similar traffic shows the similar statistical patterns at different time scales, and the ranges is from 100ms to the minutes, which means that the extended bursts of activity and inactivity still exist regardless of whether we look at millisecond, second, minute, or hour averages. This is in dramatic contrast to Poisson process which has the exponential distribution such that over long time scales, Poisson process has a characteristic burst length which tends to be smoothed by averaging over a long enough time scale.

For a stationary time series $X(t), t \in \Re$, where $X(t)$ is interpreted as the traffic volume at time instance t. The aggregated X^m of $X(t)$ at aggregation level m is defined as $X^m(k), k = 0, 1, 2, ...$, where

$$X^m(k) = \frac{1}{m} \sum_{i=km-(m-1)}^{km} X(t) \tag{6.1}$$

That is, $X(t)$ is partitioned into non-overlapping blocks of size m, their values are averaged, and k indexes these blocks. Denote $R_{X^m}(k)$ and $R_X(k)$ as the auto-correlation functions of X^m and X, respectively. $X(t)$ is called self-similar with Hurst parameter $H = 1 - 0.5\beta$ ($0.5 < H < 1$), if

$$Var(X^m) = \frac{Var(X)}{m^\beta} \tag{6.2}$$

$$R_{X^m}(k) = R_X(k) \tag{6.3}$$

where $0 < \beta < 1$. Self-similarity has two important features: heavy-tailed distribution and the multi-time scaling nature. A statistical distribution is heavy-tailed, if $P[X > x] \sim x^{-\alpha}$ where $0 < \alpha < 2$. In the heavy tail distribution, most of the observations are small, but most of the contribution to the sample mean or the variance comes from the few large observations [68]. The Pareto distribution is a simple heavy tailed distribution with probability mass function defined as

$$p(x) = \alpha k^\alpha x^{-\alpha-1}, 1 < \alpha < 2, k > 0, x \geq k \tag{6.4}$$

where α is the tail index, and k is the minimum value of x. Multi-time scaling nature means that $X(t)$ and its time scaled version $X(at)$, after normalization, must follow the same distribution. That is, if $X(t)$ is self-similar with Hurst parameter H, then for any $a > 0, t > 0$,

$$X(at) =_d a^H X(t) \tag{6.5}$$

where $=_d$ stands for equality of first and second order statistical distributions and a is called the scale factor.

The variance-time plot and R/S plot are two of the commonly used methods to calculate the Hurst parameter H. For a given set of observations $X(t) = x_t$, where $t = 0, 1, 2, \cdots$, the sample mean is:

$$M(N) = \frac{1}{N} \sum_{j=1}^{N} X_j \tag{6.6}$$

and the sample variance is S. The rescaled adjusted range (R/S) statistic is given by

$$R/S = \frac{\max_{1\leq j\leq N}[\sum_{k=1}^{j}(X_k - M(N))] - \min_{1\leq j\leq N}[\sum_{k=1}^{j}(X_k - M(N))]}{\sqrt{\frac{1}{N}\sum_{j=1}^{N}(X_k - M(N))^2}} \tag{6.7}$$

R/S plot to calculate H is

$$R/S \sim (N/2)^2 \tag{6.8}$$

6.2 Self-similar Traffic Aggregation

The overall traffic on Internet can be regarded as the aggregation of the traffic from individual traffic sources. The research work in [73] showed that as the number of traffic sources increases, the resulting traffic remain bursty. In [69], the simulation results also showed that the aggregation or superimposition of internet traffics on optical switch does not result in a smoother traffic pattern. Actual network traffic may be the aggregation of different traffic streams having different characteristics. It was shown in [69] H of the aggregated traffic is roughly equal to the maximum of the Hurst parameters of the individual self-similar sources. Here the interest focuses on the aggregated bursty traffic on the edge devices of high speed networks, such as edge routers and edge switches, for two reasons. First, as compared to the traffic nearby the source node, the high-speed aggregated traffic at the edge network devices is relatively more stable. Secondly, the aggregated traffic is easy to be extracted from the whole network at those edge devices, without much influence on the whole network's performance.

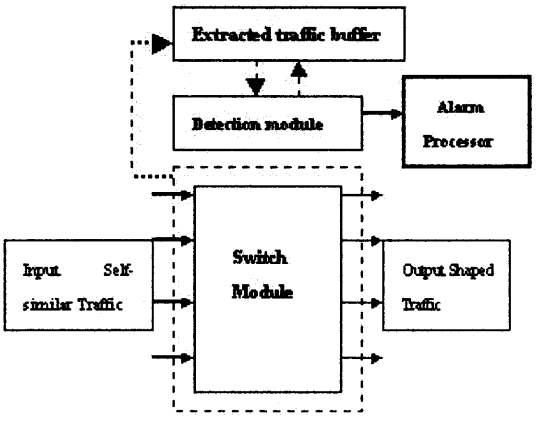


Figure 6.2 Self-similar traffic aggregation.

Figure 6.2 shows the functional architecture of an edge switch of an high speed network. It includes a switch module, queue buffer pools, monitor module, and the inlet and outlet links. The input traffic is aggregated at inlet links whereas the output aggregated traffic appears at the output side of the switch module. Each outlet link has a queue buffer pool, which is composed of several FIFO buffers. A buffer is assigned for each QoS class. Several incoming packets with the identical source address, destination address and QoS class are buffered into one burst. The monitoring module extracts the input aggregated traffic and split it into some data sets. Meanwhile, the traffic anomaly detection model detects the traffic anomaly. The monitoring module forwards the detection results to the alarm processor.

A burst assembly mechanism is reported in [69]. However, this method only assumes a constant timeout value. The scheme adopts the dynamic timeout value. Suppose the $j^{th}(j = 1, \cdots m)$ buffer pool is assigned for the j^{th} outlet link of the switch module. Each buffer pool is composed of n QoS buffers. The switch fabric in the device transfers every incoming packet to the q_i buffer of a certain outlet. If the buffer of the buffer pool is not overflowed or timeout $T_{i,j}$ has not expired, the incoming packet is buffered. Otherwise, the buffered packet is sent out and $T_{i,j}$ is reset. Two triggers for sending out the packet in the q_i buffer of every buffer pool are the buffer overflow and the timeout expiration. Timeout $T_{i,j}$ is started whenever a packet arrives in the empty q_i buffer of the j^{th} buffer pool. Since the network traffic is not stable, the number of timeout triggers and the number of the buffer overflow triggers for the i^{th} buffer in the j^{th} buffer pool are recorded as $N_{(i,j)timeout}$ and $N_{(i,j)buffer\ overflow}$. Let $P_{(i,j)timeout}$ be the percentage of the number of timeout triggers with respect to the total triggers in every i^{th} buffer in the j^{th} buffer pool. $P_{(i,j)timeout}$ is defined as:

$$P_{(i,j)timeout} = \frac{N_{(i,j)timeout}}{N_{(i,j)timeout} + N_{(i,j)buffer\ overflow}} \tag{6.9}$$

The value of timeout $T_{i,j}$ can be determined based on $P_{(i,j)timeout}$. For example, if $P_{(i,j)timeout} = 0.6 > 0.5$, which means that the traffic load is not high, the timeout value should increase, and vice versa. Thus, let timeout

$$T_{i+1,j} = \begin{cases} 2\ T_{i,j} P_{i,j}\ Timeout, & if P_{i,j}\ Timeout > 0.5 \\ \frac{1}{2} T_{i,j}, & otherwise \end{cases} \tag{6.10}$$

For example, in the algorithm the buffer size b is 2560 bits, the packet data length is constant, the initial timeout T_0 is 0.001s, and the simulation time is four seconds. The destination outlets are randomly chosen. In [72], six incoming traffic traces were generated, and these traces were aggregated at the edge switch, as shown in Figure 6.3. After going through the queuing FIFO buffers, shaped aggregated traffic was forwarded to the high speed network. Table 6.1 shows the assumed Hurst parameters, and the measured Hurst parameters of these traces and aggregated trace. Input aggregated traffic H is 0.8543, which approaches to the maximum H value of the six input traces. The output aggregated traffic H is 0.8335, which means that simply aggregating self-similar traffic traces at the edge devices will not decrease burstiness. However, after going through the traffic shaping, H decreases to 0.771348.

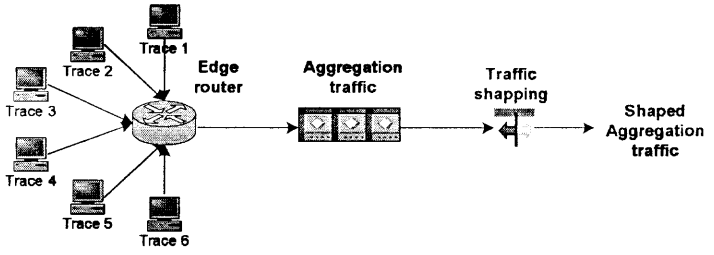


Figure 6.3 Traffic aggregation at edge router.

Table 6.1 Six Input Traces for Simulations

Input traces	Assumed H	Variance-time H	R/S H
Trace 1	0.55	0.5323	0.5225
Trace 2	0.6	0.5881	0.5775
Trace 3	0.65	0.6038	0.6356
Trace 4	0.8	0.8073	0.8011
Trace 5	0.9	0.8728	0.8641
Trace 6	0.9	0.8623	0.8357
Input aggregated	0.9	0.8543	0.8295
Output aggregated	0.9	0.8335	0.8054
Shaped aggregated	-	0.7713	0.7000

6.3 Histogram Feature Vector

In this section, HFV of self-similar traffic is introduced based on two important features of self-similarity: the multi-time scaling nature and heavy-tailed distribution. A heavy-tailed distribution gives rise to large values with non-negligible probability so that sampling from such a distribution results in the bulk of small values but with a few large values. As a result,

HFV of bursty traffic presents the left skewness characteristic. Multi-time scaling nature means that self-similar trace $X(t)$ and its aggregated traffic at time scale "at", $X(at)$, follow the same first order and second order statistical distribution. Based on $X(at)$, the reference trace $X_r(t)$ can be generated:

$$X_r(t) = a^{-H} X(at) \tag{6.11}$$

Afterwards, $X(t)$ is divided into k bins, and HFVs of these bins and $X_r(t)$ are calculated. The optimum bin number k has an important effect on HFV. If k is too small, it will make the histogram over-smoothed, while an excessively large k will not correctly reflect the traffic changes. Sturges [10] used a binomial distribution to approximate the normal distribution. When the bin number k is large, the normal density can be approximated by a binomial distribution $B(K-1,\ 0.5)$, and the histogram frequency is $\binom{a}{k}$. Then, the total number of data is

$$n = \sum_{i=0}^{k-1} \binom{k-1}{i} \tag{6.12}$$

According to the Newton's Binomial Formula:

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k \tag{6.13}$$

Let $x = 1$ and $y = 1$, then have

$$n = \sum_{i=0}^{k-1} \binom{k-1}{i} = 2^{k-1} \tag{6.14}$$

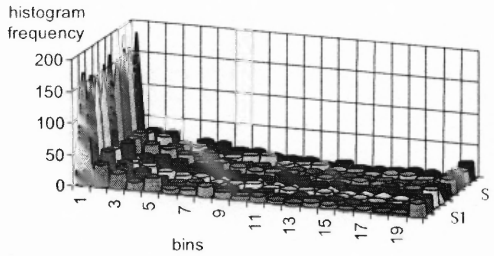
Therefore, the number of bins to choose when constructing a histogram is

$$k = 1 + \log_2 n \tag{6.15}$$

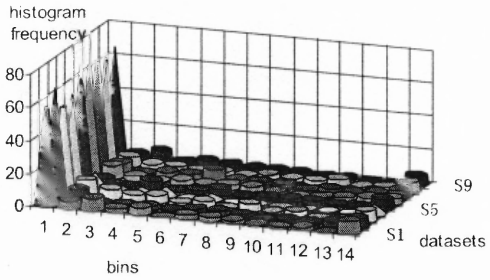
Doane [74] modified Sturges' rule to allow for skewness. The number of the extra bins is:

$$k_e = \log_2(1 + \sqrt{\frac{\sum_{i=1}^{n} (x_i - \mu)^3}{(n-1)\sigma^3}}) \tag{6.16}$$

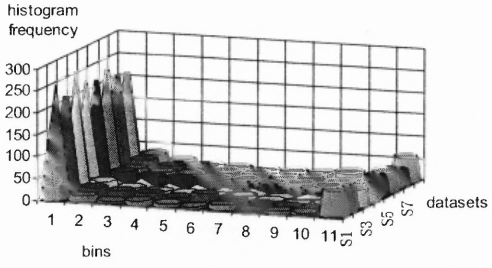
However, because of the normal distribution assumption, Sturges' rule and Doane still do not always provide enough bins to reveal the shape of severely skewed distribution (i,e.,



(a) HFV of traffic trace 4.



(b) HFVs of 1999 DARPA traffic trace.



(c) HFVs of aug89 traffic trace.

Figure 6.4 Traffic HFVs with reference trace.

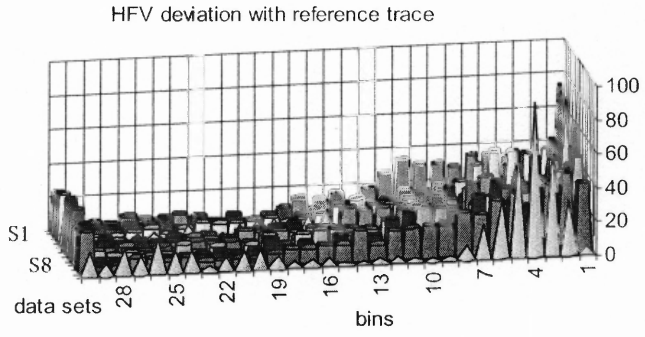
self-similarity). Therefore, they often over-smooth the histograms. The Doane's equation is modified to:

$$k = 1 + \log_2 n + 4H \log_2(1 + \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^3 n}{(n-1)\sigma^3}}) \tag{6.17}$$

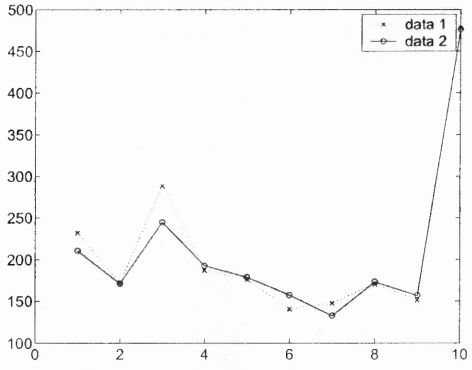
In the simulations, the traffic traces come from various sources: traffic trace 4 mentioned in Section 2, the aug89 trace from [8], and DARPA 1999 week 2 Friday data set from [50]. Figure 6.4 shows HFVs of these data sets with the left skewness distributions. Due to the heavy-tail distribution, a large number of small values locate in the first three bins, whereas few large values locate in the remaining bins (the relative high histogram frequency in the last bin is caused by the large bin size).

6.4 Traffic Anomaly Detection

In this section, Multi-Time scaling Detection (MTD) is proposed to detect the self-similar traffic anomalies by comparing the distribution deviation between traffic sets with reference trace. In particular, the interest focuses on the self-similar aggregated traffic on the edge network devices. As a large volume of traffic flows through these network access points at very high speed, these access points are most suitable for placing the anomaly detection systems. A segment packet inter-arrival time trace $X(t)$ of the input aggregated traffic at an edge switch



(a) Traffic HFVs.



(b) Deviations from reference model.

Figure 6.5 Detect anomaly traffic with MTD.

includes the abnormal traffic. From (6.17), total bin number is set to 30, and the aggregate traffic in time scale $30t$, $X(30t)$, is built up. The reference trace is $X_r(t) = 30^{-H}X(30t)$, and the whole time series is divided into 30 data segments. For each segment, the histogram frequency is calculated, and HFV includes 30 histogram frequencies. MTD compares the HFV of $X(t)$ with $X_r(t)$ by deviation. Figure 6.5(b) presents the HFV of the last 10 bins (21^{th} to 30^{th}). The traffic anomaly exists in the 30^{th} bin, with sharp distinction. Figure 6.5(a) shows that the traffic anomaly exists in the 30th data set, which is distinctly different from the reference model.

In the simulation, it is observed that each data set appears to have different burstiness characteristics, which should be considered when detecting traffic anomalies. Smoothed burstiness parameter β and the burstiness compensation parameter γ are used to smooth the deviations error, which results from the burstiness of data set. The smoothed burstiness parameter of the i^{th} data set is defined as

$$\beta_i = \frac{\mu\ of\ largest\ 1\%\ observation}{\mu\ of\ whole\ data\ set} \tag{6.18}$$

Based on β_i, the bursty compensation parameter γ_i of the i_{th} data set is defined as

$$\gamma_i = \frac{\beta_{mean}}{\beta_i} \tag{6.19}$$

and the compensation weight ω_i is:

$$\omega_i = \sum_{j=1}^{k} \left(\frac{(x_j \gamma_j - Z_j)^2}{Z_j} - \frac{(x_j - Z_j)^2}{Z_j} \right) \tag{6.20}$$

where k is the total number of bins of the histogram, and Z_j is the number of packets in the j^{th} bin of the histogram. Table 6.2 shows the HFV deviations error with compensation. ω smoothes the deviation error caused by the burstiness of the traffic within the well-behaved data sets; it has little effect on that in the abnormal traffic.

Table 6.2 Smooth Deviations Error with Compensation

Bin #	21	22	23	24	25	26	27	28	29	30
β	6.43	5.61	6.66	4.89	4.69	5.02	6.81	4.83	4.81	4.92
γ	0.85	0.97	0.82	1.16	1.16	1.09	0.81	1.13	1.15	1.11
ω	-20.5	0	-43.8	+5.28	+2.52	+16.5	-14.9	+2.6	+5.6	-2.34

In this chapter, MTD analyzed the aggregated traffic to identify traffic anomalies. The monitoring module generates the reference traffic model, and can detect traffic anomalies by large anomaly deviation from ordinary-behaving traffic with the compensation weight.

CHAPTER 7

CONCLUSIONS

7.1 Summary

This dissertation presents an IDS framework for extracting the attack scenarios from log data for attack reasoning and query. In addition, the current research goals for IDS have been addressed. This dissertation expounds on the research efforts in applying NLP and FOL based methods in implementing the FAR-FAR semantic scheme. The following aspects have been addressed in this dissertation:

- In Section 2, inspired by the semantics of attack behaviors, the work in this dissertation combined the NLP and FOL ideas in IDS. The layered attack knowledge representation formalism and the functional architecture of the semantic scheme were also reported.

- The basic concepts of DL was introduced in Section 3. The modified case grammar, PCTCG was used to transfer the heterogeneous alerts into machine-understandable uniform streams. Furthermore, a robust Variant Packet Sending-interval Link Padding algorithm (VPSLP) was proposed to prevent the links between the IDS sensors and the FAR-FAR agents from traffic analysis attacks.

- In Section 4, the alert semantic network, 2-AASN, was used to generate the attack scenarios to alert the security administrator. Then, the correlation operations and rules were defined to extract the alerts correlations from 2-AASN. ACW was also considered when extracting attack strategies to decrease the false alarm rate and real-time processing.

- In Section 5, a conjunctive semantic query approach was presented to query the attack status. This semantic attack knowledge query language was based on the technique of Spreading Activation.

- In Section 6, the bursty traffic anomaly detection method, Multi-Time scaling Detection (MTD), was proposed to statistically analyze the network traffic's Histogram Feature Vector to detect the traffic anomalies.

7.2 Future Research

The work addressed in this dissertation is not near to be conclusive. This dissertation focuses on the requirements for IDS from the viewpoint of alert fusion and high-level reasoning. In order for the framework to be as comprehensive as possible, more work need to be done. The future research directions include:

- The action-centric ontology have developed. However, the design and enrichment of the attack ontology or taxonomy requires further development.
- The attack prediction model should be implemented to detect early stages of an attack, based on the categorization results of the alerts.
- Currently, the weights of the ontology relations are set manually according to the security level of the attack actions. In order to meet the requirements of different security administrators, a new algorithm is needed to calculate those weights dynamically.
- The techniques for enabling communication among IDS sensors and FAR-FAR needs further study.

REFERENCES

[1] P. Kazienko and P. DOROSZ, "Intrusion detection systems (ids) part i - (network intrusions; attack symptoms; ids tasks; and ids architecture)," *http://www.windowsecurity.com*, Apr, 2003.

[2] H. Kung, C. Cheng, K. Tan, and S. Bradner, "Design and analysis of an ip-layer anonymizing infrastructure," *Proceedings of the Third DARPA Information Survivability Conference and Exposition*, pp. 62–75, Washington D.C., April, 2003.

[3] E. Amoroso, *Intrusion Detection: An Introduction to Internet Surveillance, Correlation, Trace Back, Traps, and Response.* Intrusion.Net Books, 1999.

[4] Dragon. *http://www.enterasys.com/products/ids/.*

[5] ISS. *http://www.iss.net.*

[6] Snort. *http://www.snort.org.*

[7] EMERALD. *http://www.sdl.sri.com/emerald.*

[8] NADIR. *http://nadir.lanl.gov/.*

[9] NIDES. *http://www.csl.sri.com/nides.*

[10] Cisco. *http://www.cisco.com/warp/public/cc/pd/sqsw/sqidsz/.*

[11] Symantec. *http://enterprisesecurity.symantec.com/products.*

[12] C. Grace, "Understanding intrusion detection systems," *PC Network Advisor*, vol. 122, pp. 11–15, 2000.

[13] F. Cuppens and A. Miège, "Alert correlation in a cooperative intrusion detection framework," *Proceedings of 2002 IEEE Symposium on Security and Privacy*, pp. 187–200, Oakland, USA, September, 2002.

[14] S. Manganaris, M. Christensen, D. Zerkle, and K.Hermiz, "A data mining analysis of rtid alarms," *Computer Network*, vol. 34 n.4, pp. 571–577, Oct. 2000.

[15] C. King, "Sem: Navigating the seas of security event data," *Network Magazine*, May, 2004.

[16] D. Curry and H. Debar, "IDMEF," *http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-03.txt.*

[17] H. Debar and A. Wespi, "Aggregation and correlation of intrusion-detection alerts," *Proceedings of the 4th International Symposium, Recent Advances in Intrusion Detection*, pp. 85–103, Davis, CA, USA, October, 2001.

[18] O. M. Dain and R. K. Cunningham, "Building scenarios from a heterogeneous alert stream," *Proceedings of IEEE Workshop on Information Assurance and Security*, pp. 5–6, West Point, NY, June, 2001.

[19] A. Valdes and K. Skinner, "Probabilistic alert correlation," *Proceedings of Workshop on Recent Advances in Intrusion Detection*, pp. 54–68, 2001.

[20] P. Ning, Y. Cui, and D. S. Reeves, "Constructing attack scenarios through correlation of intrusion alerts," *Proceedings of the 9th ACM Conference on Computer andCommunications Security*, pp. 245–254, Washington D.C., 2002.

[21] W. Schwartau, *Time Based Security.* Inter Pact Press, 1999.

[22] ACID. *http://acidlab.sourceforge.net.*

[23] J. Undercoffer, A. Joshi, and J. Pinkston, "Modeling computer attacks: An ontology for intrusion detection," *Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection*, pp. 113–135, Pittsburgh, PA, USA, 2003.

[24] J. Howard and T. A. Longstaff, "A common language for computer security incidents," *Sandia National Laboratories technical report*, vol. SAND98-8667, 1998.

[25] B. Moring, L. Me, H. Debar, and M. Ducass, "M2d2: A formal data model for ids alert correlation," *Proceedings of the 5th International Symposium, Recent Advances in Intrusion Detection*, pp. 115–137, Zurich, Switzerland, Oct. 16-18, 2002.

[26] V. Raskin, S. Nirenburg, M. J. Atallah, C. F. Hempelmann, and K. E. Triezenberg, "Why nlp should move into ias," *Proceedings of the Workshop on a Roadmap for Computational Linguistics*, pp. 1–7, Taipei, Taiwan, 2002.

[27] V. Raskin, C. F. Hempelmann, K. E. Triezenberg, and S. Nirenburg, "Ontology in information security: A useful theoretical foundation and methodological tool," *Proceedings of the New Security Paradigms Workshop*, pp. 53–59, Loudcroft, New Mexico, 2001.

[28] F. Cuppens and R. Ortalo, "Lambda: A language to model a database for detection of attacks," *Proceedings of Recent Advances in Intrusion Detection*, pp. 197–216, Washington D.C., September, 2000.

[29] CERIAS. *http://www.cerias.purdue.edu.*

[30] T. Lee, "Semantic web road map," *http://www.w3.org/DesignIssues/Semantic.html*, 1998.

[31] U. Shah, T. Finin, and A. Joshi, "Information retrieval on the semantic web," *Proceedings of the Conference on Information and Knowledge Management*, pp. 461–468, McLean, Virginia, USA, November 4-9, 2002.

[32] B. Meza, C. Halaschek, I. B. Arpinar, and A. Sheth, "Context-aware semantic association ranking," *Proceedings of the First International Workshop on Semantic Web and Databases*, pp. 33–50, Berlin, Germany, September, 2003.

[33] P. Naldurg, K. Sen, and P. Thati, "A temporal logic based approach to intrusion detection," *Proceedings of the International Conference on Formal Techniques for Networked and Distributed Systems*, Madrid, Spain, September 27-30, 2004.

[34] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, *Description Logic Handbook.* Cambridge University Press, 2002.

[35] P. Neumann, *Computer Related Risks.* The ACM Press, 1995.

[36] N. Jayaram and P. Morse, "Network security: A taxonomic view," *Proceedings of the European Conference on Security and Detection*, pp. 28–30, London, April 28-30, 1997.

[37] J. Howard, *An Analysis of Security Incidents on the Internet*, vol. Carnegie Mellon University. Ph.D. dissertation, 1997.

[38] O. Jespersen, *A Modern English Grammar on Historical Principles.* George Allen and Unwin Ltd., London, 1954.

[39] R. Lees, *The Grammar of English Nominalizations.* Indiana University Press, 1966.

[40] P. Downing, "On the creation and use of English compound nouns," *Language*, vol. 53, pp. 810–842, 1977.

[41] J. Sowa and J. Zachman, "Extending and formalizing the framework for information systems architecture," *IBM Systems Journal*, vol. 31, pp. 590–616, 1992.

[42] Protege. *http://protege.stanford.edu.*

[43] W. Yan, E. Hou, and N. Ansari, "Extracting attack scenario knowledge using pctcg and semantic networks," *Proceedings of 29th Annual IEEE Conference on Local Computer Networks*, pp. 110–117, November, 2004.

[44] C. Fillmore, *Syntax and Semantics 8: Grammatical Relations.* Academic Press, New York, 1997.

[45] W. Cook, *Case Grammar Theory.* Washington, DC: Georgetown University Press, 1st ed., 1989.

[46] R. Brachman and H. Levesque, *Knowledge Representation and Reasoning.* Morgan Kaufmann Press, 2004.

[47] L. Teft, *Programming in Turbo Prolog with an Introduction to Knowledge-based Systems.* Prentice Hall Press, 1989.

[48] X. Fu and W. Zhao, "On effectiveness of link padding for statistical traffic analysis attacks," *Proceedings of the 23rd International Conference on Distributed Computing Systems*, pp. 340–346, Providence, Rhode Island, May, 2003.

[49] W. Yan, E. Hou, and N. Ansari, "Defending against traffic analysis attack with link padding for bursty traffic," *Proceedings of the 5th International Information Assurance Workshop*, pp. 46–51, West Point, USA, June, 2004.

[50] MIT. *www.ll.mit.edu/IST/ideval/data/2000/LLS_DDOS_1.0.html.*

[51] K. Fukunaga, *Statistical Pattern Recognition.* Academic Press, 1990.

[52] R. Duda and P. Hart, *Pattern Classification and Scene Analysis.* John Wiley and Sons Press, 1973.

[53] L. Vanderwende, *The analysis of noun sequences using semantic information extracted from on-line dictionaries.* Ph.D. dissertation, Georgetown University Press, Washington, DC, 1996.

[54] J. Lucassen and R. Mercer, "An information theoretic approach to the automatic determination of phonemic baseforms," *Proceedings of ICASSP*, vol. 42.5, pp. 1–4, Washington, DC, 1984.

[55] F. Smadja, "Retrieving collocations from text: Xtract," *Computational Linguistics*, vol. 19, pp. 143–177, Washington, DC, 1984.

[56] W. Martin and P. Sterkenburg, *On the processing of text corpus, Lexicography: Principles and Practice.* R. Hartmann Press, New York, 1983.

[57] K. Church, "Word association norms, mutual information, and lexicography," *Computational Linguistics*, vol. 16.1, pp. 22–29, 1990.

[58] RealSecure. *http://www.iss.net.*

[59] Whitehats. *http://www.whitehats.com.*

[60] G. Salton, A. Wong, and C. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18.11, pp. 613–620, 1975.

[61] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *Journal of Machine Learning Research*, pp. 45–66, 2001.

[62] G. Salton and C. Buckley, "Improving retrieval performance by relevance feedback," *Journal of the American Society for Information Science*, vol. 41, pp. 288–297, 1990.

[63] A. McCallum, "Bow: A toolkit for statistical language modeling text retrieval, classification and clustering," *http://www.cs.cmu.edu/ mccallum/bow*, 1996.

[64] I. Horrocks and S. Tessaris, "Querying the semantic web: a formal approach," *Proceedings of the 13th International Semantic Web Conference*, pp. 177–191, Siguenza, Spain, October 1-4, 2002.

[65] C. Rocha, "A hybrid approach for searching in the semantic web," *Proceedings of the 13th Conference on World Wide Web*, pp. 374–383, New York, NY, USA, May 17-20, 2004.

[66] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," *Proceedings of 20th International Conference of Very Large Data Bases*, pp. 487–499, Santiago de Chile, Chile, September 12-15, 1994.

[67] K. Hwang, Y. Kwok, S. Song, M. Cai, Y. Chen, R. Zhou, and X. Lou, "Gridsec: Trusted grid computing with security binding and self-defense against network worms and ddos attacks.," *Proceedings of the International Workshop on Grid Computing Security and Resource Management*, Corolado, USA, April, 2005.

[68] W. Leland, "On the self-similar nature of ethernet traffic," *Proceedings of the IEEE/ACM Transactions on Networking*, vol. 2.1, pp. 1–14, 1994.

[69] A. Ge, "On optical burst switching and self-similar traffic," *IEEE Communications Letters*, vol. 4.3, pp. 98–100, 2000.

[70] V. Paxson and S. Floyd, "Wide-area traffic: the failure of Poisson modeling," *Proceedings of the ACM Sigcomm*, pp. 257–268, London, UK, August 31-September 2, 1994.

[71] K. Park and W. Willinger, *Self-similar network traffic and performance evaluation.* John Wiley and Sons Press, 2000.

[72] E. Hou, W. Yan, and N. Ansari, "Traffic anomaly detection and traffic shaping for self-similar aggregated traffic," *Proceedings of the Conference on Information Science and Systems*, pp. 213–218, Baltimore, USA, March, 2003.

[73] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson, "Self-similarity through high-variability: Statistical analysis of ethernet lan traffic at the source level," *Proceedings of SIGCOMM*, pp. 100–113, Cambridge, Massachusetts, August, 1995.

[74] D. Doane, "Aesthetic frequency classification," *American Statistician*, vol. 30, pp. 181–183, 1976.