ABSTRACT

STATISTICAL ANOMALY DENIAL OF SERVICE AND RECONNAISSANCE INTRUSION DETECTION

by
Zheng Zhang

This dissertation presents the architecture, methods and results of the Hierarchical Intrusion Detection Engine (HIDE) and the Reconnaissance Intrusion Detection System (RIDS); the former is denial-of-service (*DoS*) attack detector while the latter is a scan and probe (*P&S*) reconnaissance detector; both are statistical anomaly systems.

The HIDE is a packet-oriented, observation-window using, hierarchical, multi-tier, anomaly based network intrusion detection system, which monitors several network traffic parameters simultaneously, constructs a 64-bin probability density function (PDF) for each, statistically compares it to a reference PDF of normal behavior using a similarity metric, then combines the results into an anomaly status vector that is classified by a neural network classifier. Three different data sets have been utilized to test the performance of HIDE; they are OPNET simulation data, DARPA'98 intrusion detection evaluation data and the CONEX TESTBED attack data. The results showed that HIDE can reliably detect *DoS* attacks with high accuracy and very low false alarm rates on all data sets. In particular, the investigation using the DARPA'98 data set yielded an overall total misclassification rate of 0.13%, false negative rate of 1.42%, and false positive rate of 0.090%; the latter implies a rate of only about 2.6 false alarms per day.

The RIDS is a session oriented, statistical tool, that relies on training to model the parameters of its algorithms, capable of detecting even distributed stealthy reconnaissance attacks. It consists of two main functional modules or stages: the Reconnaissance Activity Profiler (RAP) and the Reconnaissance Alert Correlater

(RAC). The RAP is a session-oriented module capable of detecting stealthy scanning and probing attacks, while the RAC is an alert-correlation module that fuses the RAP alerts into attack scenarios and discovers the distributed stealthy attack scenarios. RIDS has been evaluated against two data sets: (a) the DARPA'98 data, and (b) 3 weeks of experimental data generated using the CONEX TESTBED network. The RIDS has demonstrably achieved remarkable success; the false positive, false negative and misclassification rates found are low, less than 0.1%, for most reconnaissance attacks; they rise to about 6% for distributed highly stealthy attacks; the latter is a most challenging type of attack, which has been difficult to detect effectively until now.

STATISTICAL ANOMALY DENIAL OF SERVICE AND RECONNAISSANCE INTRUSION DETECTION

by
Zheng Zhang

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 Electrical Engineering

Department of Electrical and Computer Engineering

May 2004

APPROVAL PAGE

STATISTICAL ANOMALY DENIAL OF SERVICE AND RECONNAISSANCE INTRUSION DETECTION

Zheng Zhang

Dr. Constantine N. Manikopoulos, Dissertation Advisor — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Atam P. Dhawan, Committee Member — Date
Professor and Chairman of Electrical and Computer Engineering, NJIT

Dr. Sotirios G. Ziavras, Committee Member — Date
Professor and Associate Chair of Electrical and Computer Engineering, NJIT

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

Dr. Jose Ucles, Committee Member — Date
President, XPRT Solutions, Inc.

BIOGRAPHICAL SKETCH

Author: Zheng Zhang

Degree: Doctor of Philosophy

Date: May 2004

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering
  New Jersey Institute of Technology, Newark, NJ
- Master of Science in Telecommunication Engineering
  Beijing University of Posts and Telecom., Beijing, China
- Bachelor of Science in Electrical Engineering
  Zhejiang University, Hangzhou, Zhejiang, China

Major: Electrical Engineering

Presentations and Publications:

Zhang, Z. and Manikopoulos, C., "Detecting Denial-of-Service Attacks Using PDF Statistics", *to be submitted to IEEE Trans. Systems, Man and Cybernetics.*

Zhang, Z. and Manikopoulos, C., "Methods of Classifier Training for Anomaly Network Intrusion Detection in a Production Network Using Test Network Information", *to be submitted to IEEE Trans. Systems, Man and Cybernetics.*

Zhang, Z. and Manikopoulos, C., "Architecture of the Reconnaissance Intrusion Detection System (RIDS)", *submitted to the 2004 IEEE Workshop of Information Aussrance (IAW2004)*, June, 2004.

Zhang, Z. and Manikopoulos, C., "Detecting Denial-of-Service Attacks Through Feature Cross-Correlation", *in the Poster Session of the 2004 IEEE Sarnoff Symposium*, April, 2004.

Manikopoulos, C. and Zhang, Z., "Packet Anomaly Intrusion Detection (PAID)", *Proceedings of the Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection*, September, 2003.

Zhang, Z. and Manikopoulos, C., "Investigation of Neural Network Classification of Computer Network Attacks", *Proceedings of the International Conference on information Technology: Research and Education (ITRE 2003)*, August 2003.

Zhang, Z., Manikopoulos, C. and Jorgenson, J., "Representation and Reduction of Network Intrusion Detection Data", *WSEAS Transactions on Communications*, vol. 1, pp. 47-52, 2002.

Zhang, Z., Manikopoulos, C. and Jorgenson, J., "Representation and Reduction of Network Intrusion Detection Data", *Proceedings of the 6th World Multiconference on Circuits, Systems, Communications & Computers (CSCC 2002)*, July 2002.

Zhang, Z. and Manikopoulos, C., "Methods of Classifier Training for Anomaly Network Intrusion Detection in a Deployed Network Using Test Network Information", *Proceedings of the 3rd Annual IEEE Systems, Mans, Cybernetics Information Assurance Workshop (IAW 2002)*, June 2002.

Zhang, Z., Manikopoulos, C. and Jorgenson, J., "Experimental Comparisons of Binning Schemes In Representing Network Intrusion Detection Data", *The 36th Conference of Information Sciences and Systems (CISS2002)*, March 2002.

Zhang, Z., Manikopoulos, C. and Jorgenson, J., "Architecture of Generalized Network Service Anomaly and Fault Thresholds", *Proceedings of IFIP/IEEE International Conference on Management of Multimedia Networks and Services 2001*, October 2001.

Zhang, Z. and Manikopoulos, C., "Neural Networks in Statistical Anomaly Intrusion Detection", *Neural Network World, International Journal on Non-Standard Computing and Artificial Intelligence*, vol. 11, no. 3, pp. 305-316, 2001.

Zhang, Z., Manikopoulos, C., Jorgenson, J. and Ucles, J., "Comparison of Wavelet Compression Algorithms in Network Intrusion Detection", *Proceedings of The International Conference on Computing and Information Technologies (ICCIT2001)*, October 2001.

Zhang, Z., Li, J., Manikopoulos, C., Jorgenson, J. and Ucles, J., "Neural Networks Using Neural Network Classification", *Proceedings of the 5th World Multiconference on Circuit, Systems, Communications and Computers (CSCC2001)*, July 2001.

Zhang, Z., Li, J., Manikopoulos, C., Jorgenson, J. and Ucles, J., "HIDE: A Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification", *Proceedings of the 2nd Annual IEEE Systems, Mans, Cybernetics Information Assurance Workshop (IAW2001)*, June 2001.

Zhang, Z., Li, J., Manikopoulos, C., Jorgenson, J. and Ucles, J., "A Hierarchical Anomaly Network Intrusion Detection System Using Neural Network Classification", *Proceedings of 2001 WSES International Conference on Neural Networks and Applications (NNA'01)*, February 2001.

To my parents

and

my beloved wife, Xiao Shi.

ACKNOWLEDGMENT

I would like to express my sincere thanks to my advisor, Constantine N. Manikopoulos, for giving me the opportunity to work with him and leading me into this exciting field. This dissertation could not have been completed without his continuous guidance, support and encouragement. Moreover, he taught me the discipline to become a good and effective researcher.

Special thanks to Dr. Atam Dhawan, Dr. Sotirios Ziavras, Dr. Roberto Rojas-Cessa and Dr. Jose Ucles for serving on my committee, reviewing this dissertation and providing valuable suggestions.

All the fellow members of my project team and all my friends in and out of NJIT have been great sources of ideas and fun. I thank them for making my Ph.D. study productive and enjoyable.

I am forever indebted to the love and trust of my family. My parents, my sister and my wife have always been a source of inspiration, support, advice and happiness without which I would not have been able to go this far. Words are not enough to express how much I need to thank you.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Background in Intrusion Detection

The ubiquity of the Internet allows attackers to pose serious threats on the security of computer infrastructures and the integrity of sensitive data. Computer-based intrusions in the form of a series of malicious activities typically target computer systems or the services that a host provides. The attackers often aim to obtain unauthorized privileges or to downgrade or block the availability of the services. Computer attacks may be classified into the following categories [1]:

Reconnaissance Attacks are actions initiated by attackers to probe a victim network for vulnerable servers and possible penetrating points.

Denial-of-Service Attacks are destructive attempts to interrupt or degrade the services provided by the system so that legitimate users are denied from accessing these services.

Privilege Escalation Attacks are attacks through which an attacker illegally escalates his privilege level to access and control the victim system.

Data Intercept/Alternation Attacks are activities that aim to intercept and alter sensitive data without the authorizations to do so.

System Use Attacks are actions to hijack the victim system for some other unauthorized usage, such as converting the host into an FTP server to store pirated music, or using the system as a staging point to launch attacks on other systems.

Intrusion detection systems (IDSs) are designed to automatically detect these malicious activities against a computer or a computer network.

Intrusion detection has been an active field of research for more than two decades since James Anderson published his ground-breaking paper about computer security in 1980 [2]. In 1987, Dorothy Denning [3] laid the methodological framework to detect computer-based intrusions. The basic assumptions of intrusion detections are:

- The behaviors of users and programs are observable and can be modeled from various types of audit data, for example, from system logs or from network traffic;
- Moreover, by finely tuning the system, intrusive activities can be differentiated and identified from normal activities.

Intrusion detection techniques can be broadly partitioned into two complementary approaches: *misuse* detection, and *anomaly* detection. Misuse detection systems, such as [4,5], model the known attacks and scan the audit data for the occurrences of these specific patterns. Anomaly detection systems, such as [6,7], flag intrusions by observing significant deviations from typical or expected behavior of the system or users.

The majority of the intrusion detection systems are developed following either one or the both of these two approaches. For example, SNORT [8], Bro [9] and JAM [5] are misuse detection systems; IDES [10] and INBOUND [11] detect attacks based on anomalies; some other systems, such as NIDES [12] and CMDS [13], use both misuse and anomaly techniques to detect attacks.

1.2 Dissertation Approach

This dissertation describes the studies on

The application of probability density function (PDF) statistics and neural network classification in the fields of anomaly intrusion detection and alert correlation.

More specifically, this dissertation aims to answer the following questions:

- Can PDF statistics be used for anomaly intrusion detection?
- What is the achievable performance of an IDS by using PDF statistics in anomaly intrusion detection systems?
- How do different classification algorithms perform in detecting network-based attacks?

- How well can distributed, stealthy probing and scanning attacks be identified using PDF statistics and neural classifiers?

Differing from most contemporary intrusion detection systems, which model user and attacker activities using statistical counters about system utilization and the frequencies of interesting events, this dissertation proposes a novel approach to represent statistical features in the formats of Probability Density Functions (called PDF from now on), to compare the observed parameters with the reference models using PDF similarity metrics, and to classify the measured similarity distances using neural networks.

These PDF statistics and neural classification approaches are also applied to detect stealthy probing and scanning attacks, whose objectives are to gather important information, such as network topologies, services and vulnerabilities, about a victim network.

Two intrusion detection prototypes have been designed and implemented to validate this proposed "PDF/neural network" approach:

HIDE (or Hierarchical Intrusion Detection Engine) is an anomaly intrusion detection system, with hierarchical architecture, that utilizes PDF statistical models and neural classifiers to detect Denial-of-Service (*DoS*) attacks.

RIDS (or Reconnaissance Intrusion Detection System) is a session-oriented, statistical anomaly detection system that detects probing and scanning (*P&S*) reconnaissance attacks. It consists of two main function modules: the reconnaissance activity profiler (RAP) for *P&S* attack detection and the reconnaissance activity correlater (RAC) for alert and scenario correlation.

1.3 Basic Concepts

Some basic terminologies of intrusion detection, which will be extensively used throughout this dissertation, will be first introduced in this section. The criteria in evaluating intrusion detection systems are described in the following subsection.

1.3.1 Terminologies

Intrusion is "any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource" [14].

Event is the unit of analysis, or the granularity, of an IDS (see Section 2.5 for more discussions). In HIDE, an event is defined as all the packets that are observed within a time window. In RAP, an event is defined as a session state transition. In RAC, an event is defined as an alert generated by RAP.

Intrusion Detection is a process to automatically detect and alarm when an intrusion is taking place.

Attack Scenario is a sequence of attacks that an attacker launches in order to achieve certain purposes.

Alert is a warning message that an IDS generates when it finds that an intrusion is undergoing.

Alert Correlation is a procedure that automatically correlates alerts based on their similarities; it also rebuilds the attack scenarios so that more contextual and environmental information could be provided to administrators.

PDF Statistics is a set of statistical algorithms about how to represent the distribution of descriptive and representational parameters of the system under monitoring and how to statistically compare two distributions.

Detection Classifier is the algorithm that an intrusion detection system uses to classify the input data into either normal or anomalous category based on the knowledge it learned from the training data.

1.3.2 Evaluation Criteria

The performance of an intrusion detection system is generally evaluated by the following two quantities:

True Positive Rate (or detection rate) is the rate that an attack event will be detected. An ideal IDS is expected to operate at detection rates as close to 1 as possible. In the remaining of this dissertation, the detection rates are also represented as "TPR".

False Positive Rate (or false alarm rate) is the rate that normal events will be mistakenly classified as attacks. False alarm rates are expected to be close to 0. In the rest of this dissertation, the false alarm rates are noted as "FPR".

These two criteria are intertwined, and in general it is not possible to simultaneously achieve a TPR of 1 and a FPR of 0. Therefore, for IDSs with adjustable detection thresholds, Receiver Operating Characteristic curves are also used to evaluate the system performance:

Receiver Operating Characteristic curve (or ROC curve) is the curve of FPR vs. TPR at various detection thresholds. The area below the curve represents the probability of correctly distinguishing a (normal, attack) pair. A sample ROC curve is illustrated in Figure 1.1. The point at the upper left corner corresponds to the optimal detection threshold with high detection rate and low false alarm rate.

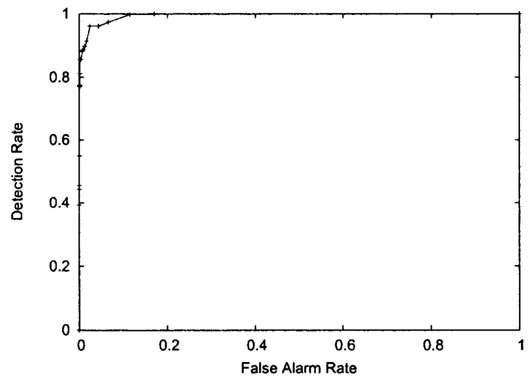


Figure 1.1 A sample ROC curve.

In this dissertation, another quantity is also used to represent the system performance.

Misclassification Rate is the probability that an observation could be mistakenly classified. It is calculated as the ratio between the number of misclassifications, including both false positives and false negatives, and the total number of observations. In the rest of this dissertation, misclassification rate is also symbolized as ERR.

The misclassification rate can be regarded as the metric describing the overall classification performance. It may be used as the objective function to train all classifiers tested in this dissertation.

1.4 Dissertation Outline

The rest of this dissertation is organized in the following way.

Chapter 2 introduces the basic concepts and the state of art in information security, intrusion detection, alert correlation and the issues on intrusion detection evaluation.

Chapter 3 describes the system architecture and the statistical model of HIDE. The classification results of HIDE on three independent test data sets are also reported in this chapter.

Chapter 4 reports the research to identify and select feature sets important for HIDE to detect *DoS* attacks.

Chapter 5 presents the systematic studies on various approaches to optimize the detection performance of HIDE, which includes the PDF partitioning schemes, the distribution similarity metrics, the classification algorithms, and the application of Wavelet compressions.

Chapter 6 presents two different methodologies to train the neural classifiers based on the attack and background traffic information collected from a controlled test network and the background traffic information collected from a production network.

Chapter 7 introduces the algorithms, the architectures and the experimental results of the proposed reconnaissance intrusion systems.

Chapter 8 summarizes this dissertation and outlines future work.

Appendix A lists the statistical features monitored by HIDE, RAP and RAC, and provides a description of each feature.

Appendix B briefly introduces the network topology, the related traffic emulation software and the attack label tools of the CONEX TESTBED network.

CHAPTER 2

INTRUSION DETECTION AND LITERATURE REVIEW

2.1 Information Security

Similar to other business and financial assets, the computation and information resources are also valuable assets of an organization. Therefore they should be appropriately protected. The main attributes of the computer and network security are commonly referred to as *CIA* [15]:

Confidentiality : to keep sensitive information from unauthorized disclosure.

Integrity : to protect sensitive information from unauthorized modification.

Availability : to prevent the unauthorized withholding of information and resources.

Josson [16] added another feature to this list with regard to the usage of information:

Accountability : to avoid the unauthorized use of computation resources.

The goal of information assurance is to prevent these four aspects of security from being violated. A computer-based intrusion is a series of malicious activities that target a computer or network system or services in order to compromise their security. Halme et al. [17] listed six general, non-exclusive approaches to anti-intrusion techniques to protect an organization from attacks:

- *Preemption*: To strike against the security threat before it has had a chance to launch its attack. This pro-active measure is difficult to practice since most of the attacks cannot be foreseen.
- *Prevention*: To preclude or significantly reduce the possibility of the success of a particular intrusion. Examples of this approach are "user identification and authorization", "access control" and "information encryption". These *prevention* approaches are necessary but far from sufficient since, as systems become ever more complex, there are always exploitable weaknesses in the systems due to various design and programming errors. The enormous bugs and the wide spreading virus and worms targeting such bugs in *Windows* operating systems are obvious examples.

- *Deterrence*: To intimidate attackers to hold off their attacks in fear of increasing risks and the negative consequences. Of course, if the protected resources are highly important, or if the perpetrations are unlikely to be caught, the attackers may not be scared off so easily.
- *Detection*: To identify intrusion attempts, so that the proper response can be evoked. It also can provide the important information of the incidents to administrator and proper authority for damage recovery and for tracing the perpetrator.
- *Deflection*: To divert an intruder to a pre-designed controlled "honey pot" so that no real damage could be caused, and to lure the intruder into believing that he has succeeded. The main difficulty for this approach is that to set up the "honey pot" realistically enough to fool an experienced attacker is far from easy.
- *Countermeasures*: To actively respond against intrusions while they are in progress. Common practices include "blocking the traffic from firewall", "disabling the user account" and, in some extreme cases, "shutting down the system temporarily". The effectiveness of the countermeasures is highly dependent on the accuracy of intrusion detections. An erroneous action on a normal user could deny the user's legitimate access to the services.

These anti-intrusion techniques may cooperate with each other to form multi-layered protection of the system resources, Figure 2.1.

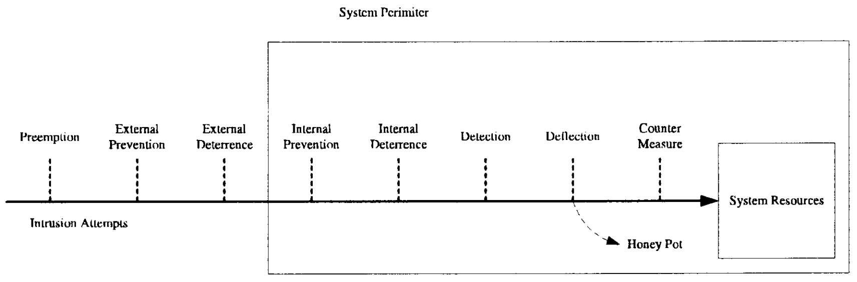


Figure 2.1 Anti-intrusion techniques. (adapted from [17])

In light of the above taxonomy, most of the current intrusion detection systems fall exclusively in the category of detection, although some systems have started to implement automated responses.

2.2 A Generic Diagram for Intrusion Detection Systems

Although intrusion detection systems may vary significantly from each other, in terms of data sources, feature extraction and classification, etc, they do share common aspects of functionality and structure among all of them. In order to tell the differences between normal and malicious activities, an IDS needs to be able to abstract the user activities to a set of statistical features. It also needs to maintain a knowledge database of known normal or attack patterns. A generic architectural model of intrusion detection systems is depicted in Figure 2.2.

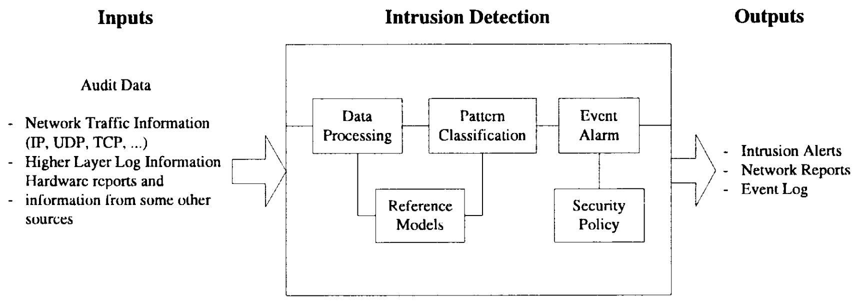


Figure 2.2 A generic diagram of intrusion detection systems.

The input data of an IDS can be from various sources: network traffic, system and application logs, network management information, etc. The input data are first processed and reduced into small sets of selected features. A database of reference models maintains the information about known attack signatures, for misuse systems, or known normal user patterns, for anomaly systems. The classification engine determines whether the data received from the data processor actually contains malicious activities. Alarms will be generated when suspicious activities are detected. Administrators may also configure the detection parameters and specify the security policies to control system reactions on different attacks.

2.3 The State of the Art in Intrusion Detection

Intrusion detection, in its essence, is a classification process, in that it tries to identify small amount of interesting attacks or anomalous patterns out of huge amount of input raw data. Therefore, the existing pattern classification techniques in other research domains, such as statistics, neural networks, pattern matching, data mining and, even, immunology, etc, had been widely borrowed and applied in this area. For example, the "association rule", which was originally proposed by Argawal et al. [18] to find out the frequent item sets in customer transaction data, had been applied to automatically discover attack detection rules in JAM [5]; The NIDES [12] used both statistical metrics and expert systems to detect anomalies and misuses; Neural Networks and Support Vector Machines (SVM), which are commonly used in the machine learning disciplines, have been used by researchers to detect attacks [19,20].

The rest of this section will briefly introduce the various intrusion detection techniques found in literature and commercial products.

2.3.1 Statistical Detection

Statistical algorithms have been widely utilized by various intrusion detection systems to extract the statistical features describing the activity patterns within the input data, to predict the expected measurements of normal/attack instances, and to compare the observed feature vectors with the expected patterns. For example, the INBOUND, [11], detects traffic anomalies by measuring the deviations of the traffic parameters from the averages of the normal users. Gao et al. [21] studied the application of Hidden Markov Models (HMMs) to profile UNIX processes. The NIDES, [12], represents user or system behaviors by a set of statistical variables and detects the deviation between the observed and the standard activities. In [22], Kolmogorov-Smirnov statistics was used to model and detect Denial-of-Service and Probing attacks.

2.3.2 Neural Networks

The neural networks are widely considered as an efficient approach to adaptively classify patterns. In [7,19], backpropagation (BP) neural networks were used to detect anomalous user activities. The Self-Organization Maps (SOMs) were applied to detect host-based attacks in [23]. Sung et al. [20] tested the importance of the prominent features in intrusion detection by using Support Vector Machines (SVMs). If properly trained, neural network classifiers can learn the mapping function between the input and output sample spaces and accurately classify unforeseen input data based on the learned knowledge. However, the knowledge that neural network classifiers acquired from training data is difficult to interpret. High computation intensity and long training cycles also hinder the applications of neural networks in real-time systems.

2.3.3 Pattern Matching Systems

Pattern matching is the most commonly used technique in misuse intrusion detection systems by searching the specific attack signatures within files or packet payloads. Example commercial systems include the RealSecure from Internet Security Systems [24], NFR from NFR Security [25]. Example open source software includes the detection engine of SNORT [8] and BRO [9]. The advantage of this technique is that this approach is straightforward and easy to be understood by administrators. However, developing and testing attack signatures are labor-intensive. It is very difficult for these systems to keep up with the evolution paces of today's attack techniques. Moreover, these systems can not detect unforeseen new attacks.

2.3.4 Data Mining Approaches

Due to the difficulties in manually developing the attack signatures and normal patterns, data mining techniques have been introduced by various researchers to automatically discover these patterns. Data mining (also known as Knowledge

Discovery in Databases - KDD) has been defined as "The nontrivial extraction of implicit, previously unknown, and potentially useful information from data" [26]. The commonly used data mining algorithms in intrusion detection systems include association rules [5], decision trees [27, 28] and clustering [29, 30], etc. These data mining techniques have drawn growing research interests since they can automatically discover detailed attack or normal models that can be easily understood by human beings. However, the drawback is that these data mining systems tend to generate a large number of models, especially for input data with large size. Extra human intervention and care must be taken to reduce and refine the extracted models.

2.3.5 Computer Immune Systems

Natural immune systems protect animals from dangerous foreign pathogens, including bacteria, viruses, parasites, and toxins. The role of the computer security systems in computers is analogous to that of the natural immune systems in animal bodies. Inspired by the principles in Immunology, Forrest and her group [31] applied the ideas from Immunology into building artificial immune systems for computers. In their approach, the problem of protecting computer systems from attacks was viewed as an instance of the more general problem of distinguishing self (legitimate users, uninfected programs) from others (unauthorized users, viruses and other malicious codes). Dasgupta et al. [32] presented another immunology-inspired intrusion detection system by using both positive (non-self) and negative (self) selection mechanisms of the immune system to detect computer attacks either in positive space (attack signatures) or in negative space (normal patterns). Although theoretically are these systems interesting, the actual effectiveness of this immunological approach is still waiting to be proved.

2.4 Attack Scenarios and Alert Correlation

An attack scenario is a sequence of attacks that an attacker launches in order to achieve certain malicious purposes. While some isolated attacks occur without warning, most network-based intrusions go through certain fairly well-defined stages to scan and compromise victim hosts. For example, the attack scenario resulting a DDoS attack, Figure 2.3, contains the following distinctive steps:

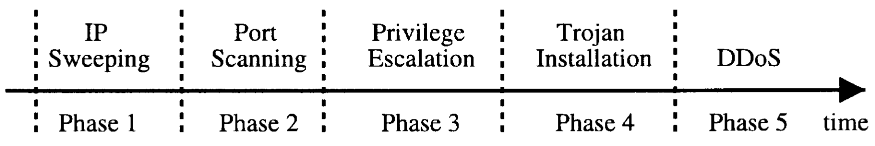


Figure 2.3 A sample attack scenario.

1. The attacker site-maps the topology of the victim network through IP sweeping;
2. Port scanning and probing attacks are launched to identify open ports, the provided services, the operating systems and the possible penetrating points and methods against the victim hosts;
3. The attacker breaks into the victim machines and gains the root privileges by exploiting these identified weaknesses;
4. Trojan and "back-door" programs are installed in the compromised systems;
5. These compromised systems are then used as attack agents to launch a distributed Denial-of-Service (DDOS) attack to another network.

The gradually unfolding characteristic of this attack scenario provides ample time for an IDS to detect and arrest it in the early stages so that the severe damages could be prevented. Detecting these different threats and discovering the attack scenarios at the earliest possible time would provide a lot of important information for system administrators to take appropriate countermeasures promptly.

Traditional intrusion detection systems are trained to detect as many suspicious activities as they could find. However, the usefulness of the generated alerts is often limited by the following two factors:

1. The positives (alerts) are in general fine-grained, low-level. Each alert corresponds to an attack fingerprint the IDS detected in a packet or a session. The arrival of the alerts is bursty and sometimes temporally overlapping. A high-level situational overview describing the whole attack scenarios is absent.
2. The "effectiveness" of current IDSs is low. "Effectiveness" is defined as the probability that a positive alert detection by an IDS is actually true. Axelsson pointed out that base-rate fallacy applies to Intrusion Detection [33]. It means, even for an IDS with very low false positive rate, the probability that an alarm indicates a real intrusion is still low, due to the fact that the population of normal activities is overwhelmingly larger than the population of intrusions.

The base-rate fallacy is one of the most serious problems for current IDSs. Every alert needs time and human power to investigate. A large number of false positives can distract the attention of administrators so that the true critical positives could accidentally be ignored. Furthermore, the number of false positives might become so high that administrators could get bored and simply ignore all alerts.

For the above reasons, the alert correlation (also called data fusion in some literature) is drawing more research interests as a solution to the above problems. The alert correlation correlates alerts of different attacks and heterogeneous ID sensors together into attack scenarios, which provides coarse-level description about intrusion plan. These scenarios could help to infer the intruders' intentions and appraise the security threats. Also, the false positives could be isolated and filtered out based on their differences in space, time and other aspects.

A number of research projects have been started in the field of alert correlation. The GrIDS, [34], uses predefined rule sets to combine alerts and network data into a graph structure to discover large scale coordinated attacks. The EMERALD, [35], correlates alerts using a probabilistic approach by comparing the similarities among the alerts. Dain, [36], proposed algorithms to estimate the probability that an alert belongs to a given scenario, and to fuse the alert with the most possible scenario. Staniford et al., [37], used simulated annealing to cluster anomalous packets together into portscan scenarios.

2.5 Intrusion Detection Evaluation

Intrusion detection is a classification process, which involves the fitting of models to data, then enabling inferences from these models. Generally, the performance of a pattern recognition system can be quantified by two measurements: the true positive rate (TPR, Probability(positive|attack)) and the false positive rate (FPR, Probability(positive|normal)). As stated in Section 1.3.2, in general, it is not possible to achieve a TPR of 1 and a FPR of 0. The common practice in evaluating pattern recognition algorithms is to choose the one with the best TPR within the constraints of an acceptable FPR.

Unfortunately, evaluating intrusion detection systems by using the above stated criteria is problematic due to the following reasons.

Problem 1 *There is no standard definition of what constitutes monitored security events.*

A typical IDS looks at a series of events and tries to identify those that represent an intrusion. The input data may be log records from one or more monitored services on a host, packets on a network, or some other descriptors of activity within the monitored domain. A security event, the unit of analysis, could be defined as a log record, a network packet, or a TCP session. This lack of universal acceptance of the event definitions could lead to big difference in TPR and FPR values, thus making comparison of systems with different event definitions meaningless.

Problem 2 *ID performance is traffic/attack sensitive.*

Intrusion detection systems are trained and tested by data collected from a test environment containing a limited number of attack instances. An IDS having good training and testing performance could yield very high FPR in another environment with completely different background settings.

Moreover, the abundance and the fast evolution on attack tools make it impossible to train ID systems with all possible attack patterns. Therefore, an IDS could have high detection rate on some kinds of attacks but perform poorly on detecting other types of attacks in a real network.

Problem 3 *Independent and unbiased data sets are hard to obtain.*

The above two stated problems could be alleviated somehow by testing IDSs with a common data set. Unfortunately independent data sets are hard to obtain, because 1) network data are hard to be properly identified (one can never be sure that there are no subtle attacks hiding undiscovered in the data); 2) background and attack models are not stationary and can not be clearly defined, thus rendering the usefulness of any static evaluation data short-lived.

2.5.1 DARPA/MIT-LL Intrusion Detection Evaluation Projects

As an effort to reliably evaluate the capabilities of existing intrusion detection systems, MIT Lincoln Labs (MIT-LL) was designated by DARPA to build a simulation network at about 1998. The background network traffic was simulated according to the traffic statistics observed in an Air Force base. A mixture of different user profiles and different attacks were simulated in the network. So far, three data sets (named as DARPA98, DARPA99 and DARPA2000) have been released and can be publicly downloaded from their web site [38].

The definitions of security events are slightly different among these three data sets. In DARPA98, an event is defined as a session, which can be TCP, UDP or ICMP. A session is characterized by a start time, duration, service, source and destination (IP address and port). Using the session as the unit of analysis is less than satisfactory since it does not provide accurate results if

- A single attack requires more than one service to be used or involves multiple sessions of the same service to be completed or

- A false alarm determination is based on data from more than one session.

In DARPA98 and DARPA2000, events are defined as attack instances, which may include multiple sessions. An attack instance is characterized by the attack date, the attack time and the victim IP address.

2.6 The Framework of EWIDS

An early warning intrusion detection system (EWIDS) is designed to detect three types of attacks: reconnaissance attacks (*P&S*), denial of service (*DoS*) attacks, and privilege escalation attacks (*R2L* or *U2R*). It also correlates the alerts of reconnaissance attacks into attack scenarios, discovers the coordinated distributed attacks, and alarms the system administrators at the earliest possible time that attacks are undergoing. A complete early warning system usually consists of the following five sub systems (see Figure 2.4):

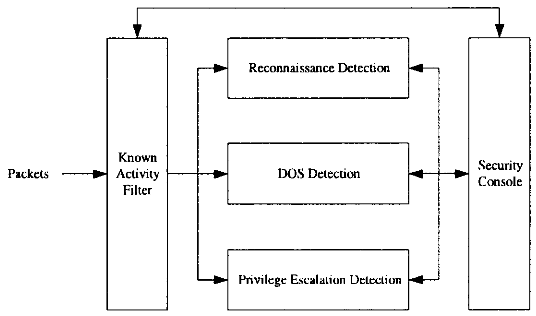


Figure 2.4 The system diagram of the EWIDS.

- Known Activity Filter (KAF): works as the packet filter for the EWIDS. It bypasses those known normal packets and drops those known attack packets. The detailed diagram of KAF is given in Subsection 2.6.1.

- *DoS* Detection System: detects *DoS* attacks at the earliest possible stage so that the attack traffic could be blocked from entering the network and significant service degradations could be avoided. The *DoS* detection system used in this EWIDS system is HIDE, which is described in Chapter 3.

- Reconnaissance Detection System: detects the reconnaissance attacks. Although reconnaissance attacks do not cause any material damage on the victim systems, they do provide the useful information about the imminence of more malicious attacks. By detecting reconnaissance activities in the earliest possible time, an early-warning system can provide the protection to thwart potential attack devastating scenarios in their early stages. The algorithms and results of the proposed RIDS system are presented in Chapter 7.

- Privilege Escalation Detection System: prevents attackers from gaining unauthorized privileges by exploiting system weaknesses. This subsystem is an integral part of a complete EWIDS but not within the scope of this dissertation.

- Security console: is the interface between administrators and the early warning system. It visualizes the attack alerts detected by the system and provides the administrator the interface to configure and to administer the whole system.

2.6.1 Known Activity Filter

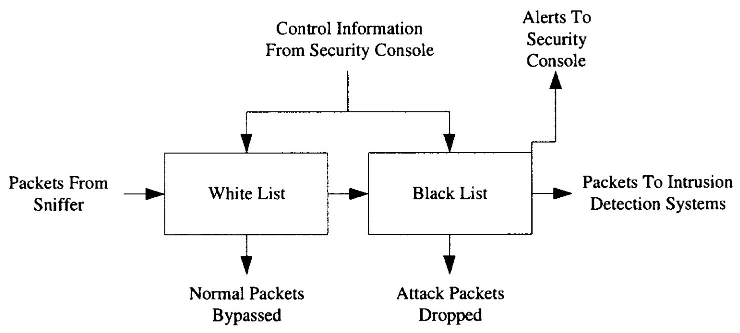


Figure 2.5 Known activity filter.

The KAF module, Figure 2.5, processes the packets sniffed from the network and filters out the packets whose source or destination addresses are listed in the back or white lists. The black list lists the IP addresses of the known attackers. Whenever KAF detects packets from these attackers, it will immediate report to security console and then drop the packets. The white list lists the IP addresses of the known normal users. By filtering out packets of known status, a great portion of computational powers can be saved to concentrate on analyzing unknown traffic.

CHAPTER 3

HIDE: A HIERARCHICAL INTRUSION DETECTION ENGINE

The HIDE [39] is a statistical anomaly intrusion detection system. It represents statistical parameters in PDF format, compares the observed parameters with the reference models using PDF similarity metrics, and classifies the measured similarity distances using neural network classifiers.

Section 3.1 briefly reviews the existing literature on *DoS* attack detection. Section 3.2 introduces the system architecture of HIDE. Section 3.3 describes the event definition and the monitored statistical features. Section 3.4 presents HIDE's statistical mode. The detection results are reported in Section 3.5.

3.1 Literature Review in *DoS* Attack Detection

The *DoS* attacks pose serious threats on the computer infrastructures of both commercial and governmental organizations. Any interrupt or downgrade of the services provided by an organization could cause huge financial loss and impair the public reputation of the organization. The recent trend of the marriage between computer viruses and *DoS* attacks (e.g. the *DoS* attacks on the web sites of Microsoft and SCO launched by the machines infected by the Mydoom virus [40]) makes this threat even more immense.

There are many varieties of *DoS* attacks. Some *DoS* attacks (for example mailbomb, neptune, or smurf attacks) abuse some legitimate features. Some other attacks (e.g. teardrop, Ping of Death) create malformed packets that target the TCP/IP protocol stacks of the susceptible machines. Still others (back, syslogd) exploit the bugs of particular network services and applications.

Most commercial and open source intrusion detection systems, such as RealSecure [24], SNORT [41] and Bro [9], etc, use both the signatures and the frequency counters of events of interest to detect *DoS* attacks. These techniques are useful in detecting known *DoS* attacks using malformed packets and exploiting specific application bugs, but they have difficult to detect new or old *DoS* attacks that abuse legitimate features.

A lot of research activities have been conducted to detect *DoS* attacks. The NIDES [12] developed sophisticated statistical algorithms to measure the distributions of short-term and long-term profiles using a χ^2-like test to measure the similarity between these two profiles. Lee et al. [5] represented network sessions using 41 various quantitative and qualitative features and utilized association rules to automatically discover the attack patterns. They processed a portion of the DARPA'98 data set using these 41 features and published the resulted data files as the KDD CUP 1999 contest data [42]. Sung [20] evaluated the importance of these 41 features using both neural network classifiers and support vector machines (SVMs). Giacinto [43] tested the effectiveness of classifying the KDD CUP 1999 data using multiple neural classifiers, one classifier for features within a category. Dasgupta [32] used immunity-based techniques and a nearest-neighbor classifier to detect the network-based attacks in the DARPA'99 data set.

3.2 The System Architecture of HIDE

The HIDE is a distributed hierarchical application, which in principle consists of multiple tiers, each tier containing several Intrusion Detection Agents (IDAs). The IDAs are intrusion detection components that monitor the activities of a host or a network. The diagram of an IDA is illustrated in Figure 3.1; it consists of the following components: probe, event preprocessor, statistical processor, neural network classifier and post processor. The functionalities of these components are described below:

- **Probe**: Collects the network traffic of a host or a network, abstracts the traffic into a set of statistical variables to reflect the network status, and periodically generates reports to the event preprocessor (see Figure 3.2).

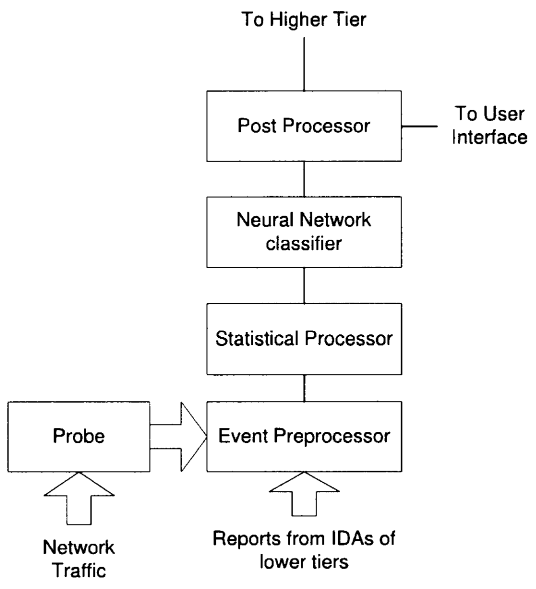


Figure 3.1 The diagram of intrusion detection agent.

- **Event Preprocessor**: Receives reports from both the probe and IDAs of lower tiers, and converts the information into the format required by the statistical model (as shown in Figure 3.2).
- **Statistical Processor**: Maintains the reference models of typical network activities, compares the reports from the event preprocessor to the reference models, and forms a stimulus vector to feed into the neural network classifiers.
- **Neural Network Classifier**: Analyzes the stimulus vector from the statistical model to decide whether the network traffic is normal or not.
- **Post Processor**: Generates reports for the agents at higher tiers. At the same time, it may display the results through a user interface.

3.3 The Monitored Events and Descriptive Statistical Features

The HIDE is a packet-oriented, time-window based intrusion detection system. The **event**, or the unit of analysis, of HIDE is defined as "the aggregated statistical pattern of all network traffic, including both incoming and outgoing packets, observed within a time window". At the end of each time window, all monitored statistical features, which represent the traffic statistics of the current time window, are extracted by

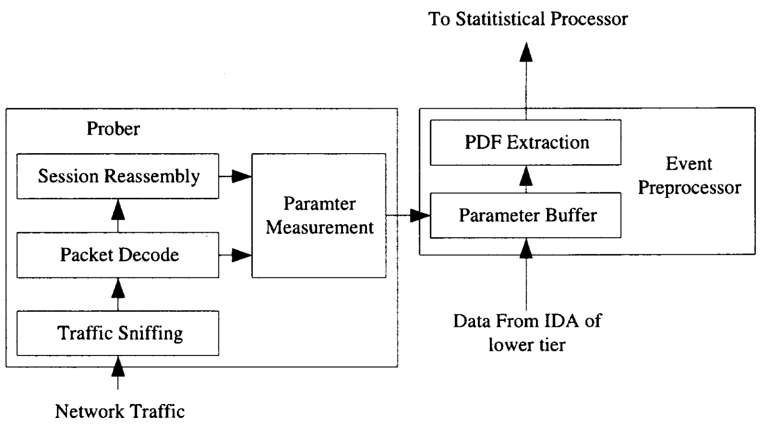


Figure 3.2 The probe and the event preprocessor.

the "Event Preprocessor" and forwarded to the following stages to detect possible attacks. The "Neural Network Classifier" will output a continuous value describing the normality of the current network traffic.

The HIDE is capable of monitoring many different statistical traffic features simultaneously to accurately describe the network status (a detailed description of all statistical features is given in Appendix A.1). However, monitoring all these features is computationally expensive and a subset of these features may already be sufficient to detect *DoS* attacks. Therefore, to achieve the maximum flexibility and efficiency, HIDE is purposefully designed to allow users to customize the statistical feature set as desired in order to monitor traffic at run-time.

3.4 The PDF Statistical Model

Statistical methods have been used in intrusion detection systems. Most of these systems simply measure the means and the standard deviations of statistical parameters and detect whether certain thresholds have been exceeded, like INBOUND [11]. SRI's NIDES [12] developed a more sophisticated statistical algorithm by using a χ^2-like test to measure the similarity between short-term and long-term profiles. In [22], Kolmogorov-Smirnov (KS) statistics was used to model and detect Denial-of-Service and Probing attacks. The current statistical model of HIDE uses a modified KS algorithm. Therefore, some basic information about χ^2 and KS statistics will first be briefly introduced in this chapter.

In HIDE, user activities are represented by a number of probability density functions. Let S be the sample space of a random variable and events $E_1, E_2, \ldots, E_K$ a mutually exclusive partition of S. Assume p_i is the expected probability of the occurrence of the event E_i, and let $p_i^{'}$ be the frequency of the occurrence of E_i during a given time interval. Let N denote the total number of occurrences.

The χ^2-like test, Equation 3.1, is used as the statistical engine of NIDES to determine the similarity between the expected and actual distributions.

$$Q = N \times \sum_{i=1}^{K} \frac{(p_i^{'} - p_i)^2}{p_i} \tag{3.1}$$

When N is large and the events $E_1, E_2, \ldots, E_K$ are independent, Q approximately follows a χ^2 distribution with $K-1$ degrees of freedom. However, in a real-time application, the above two assumptions generally can not be guaranteed, thus, empirically, Q may not follow a χ^2 distribution.

The Kolmogorov-Smirnov test, given in Equation 3.2, is applicable to measure the similarity when the underlying distribution is unknown. Assume $F(x)$ is the expected cumulative probability distribution (CDF); $F^{'}(x)$ is the measured CDF; F_i and $F_i^{'}$ are the expected and the measured CDFs at event E_i.

$$D_N = \max_{-\infty}^{\infty} \left|F(x) - F^{'}(x)\right| = \max_{i=0}^{K} \left|F_i - F_i^{'}\right| \tag{3.2}$$

The strength of the KS test is that, if $F(x)$ is continuous, the distribution of D_N does not dependent on the underlying distribution of $F(x)$, but only on the number of samples N.

Because neural networks are used as classifiers to identify possible intrusions in HIDE, it is not necessary for HIDE to know the actual distribution of D_N. However, since network traffic is not stationary and network-based attacks may have different time durations, varying from a couple of seconds to several hours or longer, a robust and efficient algorithm is needed to monitor network traffic with different time windows. Based on the above observations, a layer-window statistical model, Figure 3.3, with each layer-window corresponding to a monitoring time slice of increasing size, is chosen.

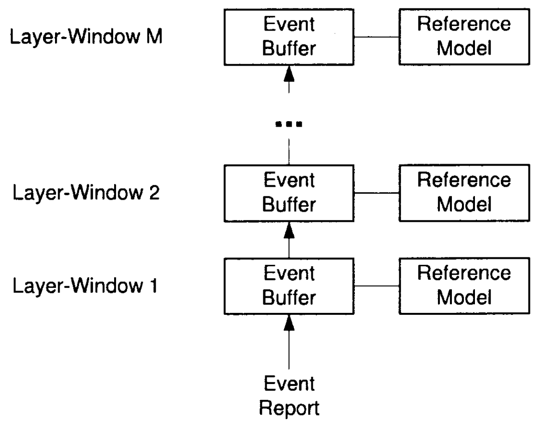


Figure 3.3 The statistical model of HIDE.

The newly arrived events will first be stored in the event buffer of layer 1. The stored events are compared with the reference model of that layer and the results are then fed into the neural network classifier to decide the network status during that time window. The event buffer will be emptied once it becomes full, and the stored events will be averaged and forwarded to the event buffer of layer 2. This process will be repeated recursively until the top level is reached where the events will simply be dropped after processing.

The similarity-measuring algorithm used in HIDE is given in Equation 3.3. The right-hand side of the equation is the combination of the KS test and the area difference between the distributions. Therefore, this equation is called the AKS test. The function $f(N)$ is a function of the number of samples N.

$$Q = f(N) \times \left[\max_{i=1}^{K}(|p_i^{'} - p_i|) + \sum_{i=1}^{K} |p_i^{'} - p_i| \right] \tag{3.3}$$

A reference updating algorithm has also been designed to update the reference model based on the newly observed real-time traffic. Let M_{old} be the reference model before updating, M_{new} be the reference model after updating and K be the observed user activity within a time window. The formula to update the reference model is given in Equation 3.4.

$$\mathrm{M}_{new} = s \times \alpha \times \mathrm{K} + (1 - s \times \alpha) \times \mathrm{M}_{old} \tag{3.4}$$

in which α is a predefined adaptation rate and s is the value generated by the output of the neural classifier. Assume that the output of the neural network classifier is a continuous variable c between -1 and 1, where -1 means intrusion with absolute certainty and 1 means no intrusion again with complete confidence. In between, the values of indicate proportionate levels of certainty. The function for calculating s is

$$s = \begin{cases} c & \text{if } c \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Through the above equations, it can be ensured that the reference model would be actively updated for typical traffic while kept unchanged when attacks occurred. The attack events will be diverted and stored, as attack scripts, for future investigation.

3.5 The Detection Results of HIDE

Three different data sets have been used to evaluate the performance of HIDE. The first data set is generated from OPNET simulations. The second data set is provided by the DARPA'98 intrusion detection evaluation project. And the third data set is collected from a testbed network setup within the CONEX laboratory of NJIT. For simplicity, the first data set will be referred as the "OPNET" data set, the second data set as the "DARPA'98" data set and the third data set as the "TESTBED" data set, here on.

3.5.1 The OPNET Simulation Data

A virtual network using simulation tools has been used to generate attack scenarios. The experimental test bed that was built using OPNET, a powerful network simulation facility, is shown in Figure 3.4. The test bed is a 10-BaseX LAN that consists of eleven workstations and one server.

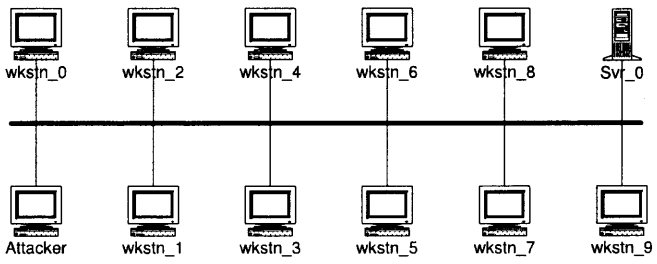


Figure 3.4 OPNET simulation testbed.

In the simulation network, three types of network traffic comprise the simulated traffic: HTTP, FTP, and SMTP (Email). To generate enough UDP background traffic, UDP is configured as the transport protocol of the SMTP services. Even though this configuration is not quite realistic, it does not undermine the general conclusions got in this experiment.

The UDP flooding attacks were simulated within the testbed. To stress-test the performance of the system, multiple simulation scenarios with different background traffic and attack traffic loads were carried out. Table 3.1 lists the traffic specifications of the simulated scenarios.

Table 3.1 Traffic Load Specifications of the OPNET Simulation

Background Traffic	Attack Traffic
600Kbps	10Kbps, 20Kbps, 30Kbps, 40Kbps, 50Kbps, 70Kbps, 100Kbps, 200Kbps
2Mbps	10Kbps, 50Kbps, 150Kbps, 200Kbps

For each simulation scenario, ten thousand records of network traffic were collected. These data were divided into two separate sets, one set of 6000 data for training and the other of 4000 data for testing.

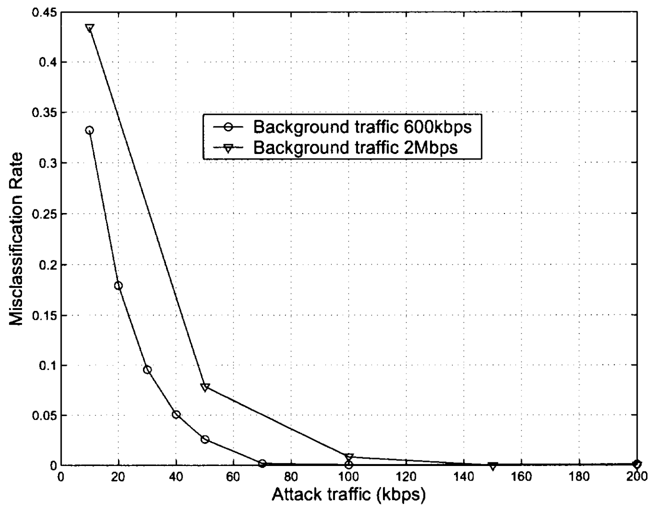


Figure 3.5 The detection results on the OPNET data.

The classification results of stress testing are shown in Figure 3.5. From the figure, the reader can see that the misclassification rates decrease as the attack level increases. This is because the traffic patterns of higher-volume attacks yield greater differences from the reference model than the differences created by lower-volume attacks. It can also be noticed that, for a certain attack level, the performance for the

600Kbps background traffic is consistently better than that of the 2Mbps background. One plausible explanation is that intruders can more easily cover their behavior patterns from being detected in high background traffic environments. Please refer to the paper [44] for more stress testing results.

Overall, the results indicate that HIDE can effectively detect UDP flooding attacks with traffic intensity as low as five to ten percent of the background intensity.

3.5.2 The DARPA'98 Intrusion Detection Evaluation Data

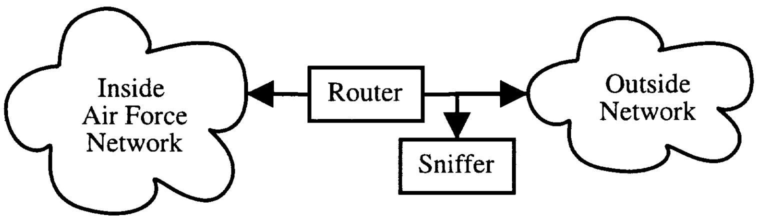


Figure 3.6 The DARPA IDEVAL network (adapted from [38]).

This data set was generated by the MIT Lincoln labs in 1998 as the intrusion detection offline evaluation project sponsored by DARPA [38]. These data were collected in an isolated controlled network environment (see Figure 3.6), which consists of two segments: the "inside" network simulates the network of an Air force base; and the "outside" network simulates the Internet. These two segments are interconnected by a router, and a sniffer is used to capture the packets passing through the router. The DARPA'98 data set includes seven weeks of training data and two weeks of testing data and contains the following attack types:

- Surveillance and Probing (*P&S*) Attacks: are used to gather information about the victim network topology, the network services, and the potential penetrating points.
- Denial of Service (*DoS*) Attacks: are attacks aiming at bringing down a server or a network so that normal users would not be able to access the services.
- Remote to Local (*R2L*) Attacks: are attacks aiming to gain unauthorized access to a local server from a remote machine.

- User to Root (*U2R*) Attacks: are used by a local unprivileged user to gain unauthorized access to local super-user privileges.

Table 3.2 Summarized Detection Results of the DARPA'98 Data Set

Total number of samples	59226
Total number of attacks	1074
Total number of misclassifications	32
Total number of false positives	25
Total number of false negatives	7
Misclassification rate	0.000540
False positive rate	0.000422
False negative rate	0.00652
"Effectiveness"	0.977

Only the network-based *DoS* attacks with observable disturbance on the network traffic are considered in the experiment. The rest attack packets are simply filtered out before entering HIDE. The *DoS* attacks considered include:

neptune is a *SYN* flooding denial-of-service attacks on one or more TCP ports to overflow TCP connection buffer through creating a large number of half-open TCP connections.

pod (or Ping of Death) is a Denial-of-Service attack to crash victim servers by sending large amount of over-sized IP packets.

Smurf is a *DoS* attack to flood the victim server by creating a large, continuous stream of ICMP "ECHO" replies.

Teardrop is a *DoS* attack that exploits the flaws of IP defragmentation programs of many older operating systems.

The detection results of HIDE are tabulated in Table 3.2. In total, 59226 useful data samples were collected. Among them, 1074 samples are attacks while the remaining 58152 samples are normal. Therefore, the base-line misclassification rate for this DARPA'98 data set is 1074/59226 = 0.0181; the latter indicates a dummy IDS that could achieve a 98.2% detection rate by simply classifying all of the attacks as normal. Table 3.2 showed that HIDE performed more than 30 times better than the base-line rate.

Table 3.3 Itemized Detection Results of the DARPA'98 Data Set

attack	samples	false negatives	FNR
neptune	798	6	0.00752
pod	24	0	0
smurf	267	1	0.00375
teardrop	12	0	0

The detection performance of HIDE on different *DoS* attacks is described in Table 3.3. From the table, it can be seen that HIDE had very low false negative rates on all the four monitored *DoS* attacks.

3.5.3 The CONEX TESTBED Data

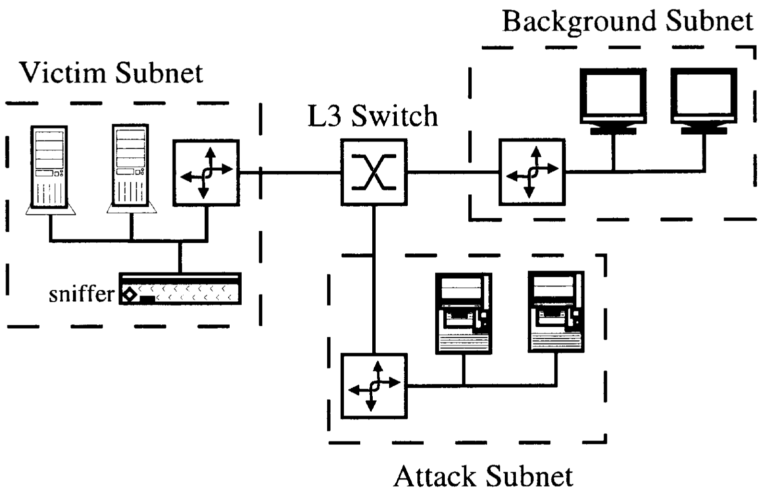


Figure 3.7 The topology of the CONEX TESTBED network.

The CONEX TESTBED network, see Figure 3.7, is a testbed network setup in the CONEX lab of NJIT as a platform to emulate network-based attacks in order to test the performance of intrusion detection prototypes. The network includes three subnets: *victim* subnet, *background* subnet and the *attack* subnet. A more detailed description about the network topology, the attack emulations and the related tools can be found in Appendix B.

Table 3.4 Summarized Detection Results on the CONEX TESTBED Data Set

Total number of samples	7582
Total number of attacks	568
Total number of misclassifications	22
Total number of false positives	18
Total number of false negatives	4
Misclassification rate	0.00290
False positive rate	0.00257
False negative rate	0.00704
"Effectiveness"	0.969

Three days worth of TCPDUMP data were collected from the CONEX TESTBED network. The detection results in testing HIDE are listed in Table 3.4. Here the base-line misclassification rate is $568/7582 = 0.0749$. As shown in the table, the misclassification rate of HIDE is about 25 times lower than the base-line error rate.

The detection performance of HIDE on detecting the five different *DoS* attacks is listed in Table 3.5. From the table the reader can see that, except for the four false negatives of the "teardrop" attacks, HIDE had perfect detection results on the other four kinds of attacks.

Table 3.5 Itemized Detection Results on the CONEX TESTBED Data Set

attack	samples	false negatives	FNR
neptune	146	0	0
pod	184	0	0
l-zonealarm	11	0	0
smurf	117	0	0
teardrop	118	4	0.0339

3.5.4 Existing DARPA Results of Other IDSs

Because the DARPA98 data sets are publicly accessible and have been widely used in various intrusion detection literatures, a natural question that may rise is how the performance of HIDE compares with other state-of-the-art intrusion detection systems.

Unfortunately, a fair comparison is difficult because the definitions of events are different between HIDE and MIT-LL evaluation projects: as stated in Section 3.2, time windows are defined as the basic units of analysis in HIDE, whereas the DARPA MIT-LL projects used "sessions" or "attack instances" as the basic events (see Section 2.5.1).

Some of the existing results when using DARPA MIT-LL data sets are briefly listed below:

1. The best *DoS* detection result in the DARPA'98 IDEVAL project can detect around "65% of all *DoS* attacks, but with very few false alarms" [45]. The DARPA'98 IDEVAL project used "session" as the unit of events.

2. The best *DoS* detection results in DARPA'99 IDEVAL project can detect about 85% of all "clear" *DoS* attacks with 10 false alarms per day (see Figure 3 of [46]). The DARPA'99 IDEVAL project used "attack instance" as the unit of events.

3. Sung [20] reported a 99.25% detection rate on *DoS* attacks by using support vector machines. "Session" is used as the basic unit in [20].

4. Dasgupta [32] detected around 96.2 percent of the selected clear *P&S* and *DoS* attacks in the DARPA'99 data set. In [32], time windows with 1-minute window size are used as basic events.

Among the listed results, Dasgupta [32] has similar definition of events as HIDE. The "99.4%" detection rate of HIDE on the DARPA'98 data set (see Section 3.5.2) is higher than that reported in [32].

The other results mentioned earlier in this section are not directly comparable with HIDE. They are listed to illustrate the performance level of the current state of the art in *DoS* detections.

CHAPTER 4

IDENTIFYING IMPORTANT HIDE FEATURES

4.1 Introduction

Detecting intrusions involves the multi-variate classifications based on the monitored features. Generally, intrusion detection systems are designed to use as many features as the designers could think they might be useful. For example, JAM [5], an intrusion detection system prototyped by Columbia University, monitors 41 different parameters for each session; HIDE (see Chapter 3) monitors 45 traffic measurements observed within a time window. The experimental results on these systems had proved that all these selected features could be used to accurately detect attack activities.

Three questions may naturally arise for these multi-variate classification approaches:

1. Are all these features equally useful?
2. If not all features are equally important, which features are more important?
3. Is it possible to remove some of the features without performance deterioration?

The issue of the Feature Subset Selection (FSS) has been widely studied in the various fields, such as Machine Learning [47], Pattern Recognition [48], Statistic [49] and Data Mining [50], etc. Generally, the researches in those areas address the FSS problems via two major models:

1. **Filter Model** [51] ranks and selects the features by measuring the statistical properties and the discrimination power of the feature subset. The commonly used filtering methods are the correlation criterion, the single feature classification, the information theoretic ranking and the Fisher criterion score.

   The "correlation criterion" [52] ranks the features based on the absolute values of the correlation coefficients between the features and the target; the "information theoretic ranking" method [53] selects features according to the mutual information between each variable and the target; the "Fisher criterion[1]", [52], scores features by calculating the statistical difference of an individual feature in positive and negative classes; the "single feature classification" [54] selects features according to the predictive power of the individual features.

2. **Wrapper Model** [51] searches the feature subset by using the learning algorithm of interest as a black box to score subsets of features based on their classification performance. Liu et al. [50] surveyed the commonly feature searching strategies in the wrapper model: *complete* search, *heuristic* search and *no-deterministic* search.

   The *complete* search strategy evaluates all possible combinations of the feature set. This strategy guarantees to find the optimal feature subset, but quickly becomes computationally impractical when the number of features grows large; the *heuristic* search strategy avoids the needs of *complete* search by employing heuristics in conducting search. This method is more computationally efficient than the complete search, but it could result sub-optimal feature subset if it's stuck in the local maxima; the *no-deterministic* search strategy tries to escape from local maxima by introducing randomness into the searching process.

The filter model is faster and can be used as a generic method for feature selection, without relying on a specific learning method. On the other hand, the wrapper model is slower but it's capable of finding an optimal or suboptimal feature subset for a given learning method.

The objective of this chapter is to identify important HIDE features, not necessary to be optimal, and to evaluate the performance of HIDE using different feature subsets. Therefore, three different feature subsets are constructed using the filtering methods:

1. *Single feature classification on all attacks.* This method evaluates the classification performance of individual features on all attacks that HIDE detects. These results are then compared with the base-line classification rates (please refer to Section 3.5.2 for the definition of base-line classification rates). The features, which perform better than the base-line rate, are regarded as important/relevant features and selected into the first feature subset.

2. *Single feature classification on individual attacks.* This method evaluates the classification performance of individual features when classifying different

[1]$F(r) = \left|\frac{\mu_p-\mu_n}{\sigma_p^2+\sigma_n^2}\right|$, where μ_n and σ_n^2 are the mean and the variance of feature r in the negative class; μ_p and σ_p^2 are the mean and the variance of feature r in the positive class

attacks, one attack at a time. The relevant features (features that perform better than the base-line rates) are selected into the second feature subset.

3. *Feature selection using decision tree.* This method utilizes a decision tree, which is capable of automatically selecting the features based on the information gain ratio [53]. The features selected by the decision tree are used as the third feature subset.

Three neural network classifiers will be trained using the three above-selected feature subsets. The classification performance will be compared with the HIDE performance, when using the full feature set (see Section 3.5).

4.1.1 Related Work in Intrusion Detection

Although all intrusion detect systems involve multi-feature classifications, surprisingly few papers have been published to systematically study the FSS problems in intrusion detection systems. Lee et al. [5] proposed an automated process to discover important features using association rules and frequent episode. Sung et al. [20] ranked the 41 features provided in the KDD Cup 99 data set [42] using support vector machines (SVMs) and neural networks.

4.2 Objective Functions

The objective function is a cost measurement function to evaluate the goodness of a particular feature subset. The commonly used objective functions in the classification problems are the measurements of the classification/prediction accuracy, such as mean-square root error, misclassification rate, false positive rate and false negative rate. Sometime, the number of features (to be minimized) is also used as a regularization term in the functions.

In this chapter, six different objective functions are devised to study their impacts on the FSS process. The first objective function, see Equation 4.1, measures the misclassification rate of a feature subset.

$$J_1(s) = \frac{N_{\text{ofp}}(s) + N_{\text{ofn}}(s)}{N} \tag{4.1}$$

Where s is a given feature set; N is total number of samples; $N_{\mathrm{ofp}}(s)$ is the number of false positive outputs of a classifier using the feature subset s; $N_{\mathrm{ofn}}(s)$ is the number of false negative outputs of a classifier using the feature subset s.

The second objective function, see Equation 4.2, takes the size of the feature subset, $|s|$, into consideration as a cost factor.

$$J_2(s) = J_1(s) + J_1(s)\log_\alpha |s| \tag{4.2}$$

In the equation, the second RHS term measures the related computation cost of the feature subset s, which is a product of $J_1(s)$ and $\log_\alpha |s|$ to avoid those trivial feature subsets with high misclassification rates and small numbers of features.

α is a scaling constant determined by the maximum computation cost, which is specified by users. For example, if $\max|s|$ is the size of the full feature set, and if the maximum computation cost is set to c, α can be calculated using Equation 4.3.

$$\alpha = (\max|s|)^{1/c} \tag{4.3}$$

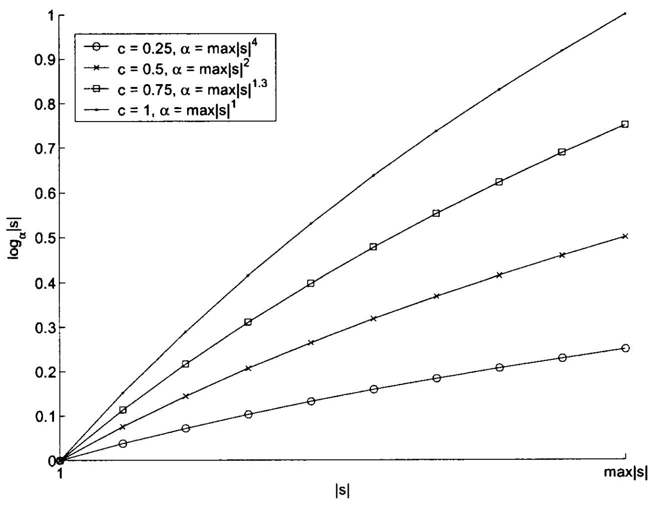


Figure 4.1 The computation cost at different α's.

Several curves of $|s|$ and $\log_\alpha |s|$ at different c's and α's are plotted in Figure 4.1 to illustrate the impact of α's on the computed costs. In this chapter, the maximum computation cost c is set to 0.25, therefore, α is set to $\max |s|^4$ accordingly.

To compare the impact of different computation cost functions, another objective function is devised to include the computation cost as a square-rooted factor, see Equation 4.4.

$$J_3(s) = J_1(s) + \beta J_1(s)\sqrt{\frac{|s|-1}{\max|s|-1}} \tag{4.4}$$

In the equation, β is a constant to modulate the computation cost. It is set to 0.25 in this chapter to conform with the maximum computation cost c, which is set to 0.25.

Because the false positives have larger impact on the performance of an IDS, three other objective functions are designed as an effort to reduce the false positives, see Equation 5 to Equation 7.

$$J_4(s) = \frac{(1+\gamma)N_{\text{ofp}}(s) + (1-\gamma)N_{\text{ofn}}(s)}{N} \tag{4.5}$$

$$J_5(s) = J_4(s) + J_4(s)\log_\alpha |s| \tag{4.6}$$

$$J_6(s) = J_4(s) + \beta J_4(s)\sqrt{\frac{|s|-1}{\max|s|-1}} \tag{4.7}$$

In the equations, γ is an adjustable constant to emphasize the importance of reducing false positives. In this chapter, γ is set to 0.6.

The feature subsets that lead to lower values of the six objective functions are desirable.

4.3 Classification using Single Feature

The performance of individual features can be measured as the lowest misclassification rate in the ROC curves, which are defined as the graphs between the true positive rates (TPRs) and the false positive rates (FPRs) at different detection thresholds [55]. The relationship between the misclassification rate, error, and the TPR and the FPR is given in Equation 4.8.

$$\text{error} = \frac{N_{\text{ofp}}(s) + N_{\text{ofn}}(s)}{N} = \frac{N_{\text{tp}}(1 - \text{TPR}) + N_{\text{tn}}\text{FPR}}{N} \tag{4.8}$$

Where N is the total number of samples; N_{tp} is the number of attack samples; N_{tn} is the number of normal samples; N_{ofn} is the number of false negatives; and N_{ofp} is the number of false positives.

The minimum misclassification rate of a feature can be calculated by examining all possible thresholds from the training data. It might seem to be computationally expensive to examine all possible thresholds, but, for a given training data with m records, there are maximally m-1 possible different thresholds. Once the training data has been sorted, the searching process can be carried out in one pass.

4.4 Experimental Results

The data files used in this chapter are the similarity distance files generated by HIDE, which monitors the 45 traffic features in the DARPA'98 data set (please refer to Appendix A for the detailed description of these 45 features). As listed in Section 3.5.2, there are totally 59226 records, including 1074 attack records and 58152 normal records, in the data files.

A *heuristic backward* search strategy is deployed in this experiment, in that it starts with the full 45 features and, after examining the performance of individual features, removes those irrelevant features.

4.4.1 Feature Filtering Based on All Attacks

Table 4.1 The Feature Ranking Table of All Attacks

Rank	Index	Name	Error
1	12	in.tcp-con-new-opened	0.00554
2	9	in.tcp-syn-pkt-rate	0.00572
3	3	in.ip-byt-rate	0.00591
4	2	in.ip-pkt-rate	0.00642
5	14	in.tcp-con-new-aborted	0.00775
6	17	in.tcp-con-diff-src	0.00864
7	13	in.tcp-con-new-closed	0.00954
8	18	in.tcp-con-diff-dst	0.00971
9	1	in.ip-pkt-len	0.00984
10	8	in.tcp-pkt-rate	0.0106
11	19	in.tcp-con-anomalous-entropy	0.0117
12	7	in.tcp-pkt-len	0.0122
13	22	in.icmp-byt-rate	0.0136
14	23	in.icmp-diff-src	0.0136
15	45	io.icmp-anomalous-echo-reply	0.0136
16	6	in.ip-csum-error-rate	0.0137
17	21	in.icmp-pkt-rate	0.0136
18	15	in.tcp-con-half-opened-ratio	0.0147
19	24	in.icmp-diff-dst	0.0156
20	16	in.tcp-con-duration	0.0170
21	11	in.tcp-rst-pkt-rate	0.0174
22	4	in.ip-frag-rate	0.0175
23	5	in.ip-defrag-error-rate	0.0176
24	28	in.udp-diff-src	0.0178
25	44	out.udp-diff-dst	0.0180
26-45	...	...	0.0181

The classification performance of these features to detect all the four kinds of attacks is evaluated and listed in Table 4.1, in which the "Error" corresponds to the misclassification rates of individual features.

From Table 4.1, it can be seen that only 25 features are relevant in DoS attack detection, the other 20 features, ranked from 26 to 45, have the same misclassification rate 0.0181, which is the base-line misclassification rate of the DARPA'98 data files. Therefore, only the first 25 useful features are selected in the feature subset. For simplicity, this feature subset is called Set_{25} from now on.

Another feature set, which includes only the best-performed feature "in.tcp-con-new-opened", is selected to illustrate the performance of the best variable. This set will be noted as Set_1 in this chapter.

4.4.2 Feature Filtering Based on Individual Attacks

This experiment examines the performance of the 45 features on individual attacks, one attack at a time.

Neptune Attack

The classification performance of individual features on *neptune* attacks is ranked and listed in Table 4.2.

Table 4.2 The Feature Ranking Table of the *neptune* Attack

Rank	Index	Name	Error
1	12	in.tcp-con-new-opened	0.00125
2	9	in.tcp-syn-pkt-rate	0.00138
3	14	in.tcp-con-new-aborted	0.00338
4	17	in.tcp-con-diff-src	0.00432
5	13	in.tcp-con-new-closed	0.00516
6	18	in.tcp-con-diff-dst	0.00545
7	8	in.tcp-pkt-rate	0.00640
8	19	in.tcp-con-anomalous-entropy	0.00748
9	7	in.tcp-pkt-len	0.00790
10	3	in.ip-byt-rate	0.00992
11	15	in.tcp-con-half-opened-ratio	0.0102
12	2	in.ip-pkt-rate	0.0105
13	16	in.tcp-con-duration	0.0127
14	11	in.tcp-rst-pkt-rate	0.0128
15	28	in.udp-diff-src	0.0131
16	44	out.udp-diff-dst	0.0133
17-45	...	...	0.0135

Because there are 798 neptune attack samples within the DARPA'98 data set, the base-line misclassification rate is 798/59226 = 0.0135. From Table 4.2, it can be seen that only 16 features, out of the full 45 features, have misclassification rates lower than the baseline rate.

Moreover, because the misclassification rates of the last two relevant features in the table, in.udp-diff-src and out.udp-diff-dst, are very close to the baseline rate, and because these parameters are monitoring the UDP traffic, which intuitively should not be closely correlated to TCP traffic, these two parameters are also not selected.

The 14 selected features are highlighted in bold fonts in Table 4.2.

Smurf Attack

The classification performance of HIDE features on smurf detection is listed in Table 4.3. Because there are 267 smurf attack samples in DARPA'98 data, the baseline rate is 267/59226 = 0.00451.

Table 4.3 The Feature Ranking Table of the *smurf* Attack

Rank	Index	Name	Error
1	22	in.icmp-byt-rate	0.000354
2	23	in.icmp-diff-src	0.000354
3	45	io.icmp-anomalous-echo-reply	0.000354
4	6	in.ip-csum-error-rate	0.000388
5	21	in.icmp-pkt-rate	0.000388
6	24	in.icmp-diff-dst	0.002126
7-45	...	...	0.00451

From Table 4.3, it can be seen that only six parameters have misclassification rates lower than the baseline rate. Therefore, these six parameters will be selected in the new feature subset.

Pod and Teardrop Attacks

Because the *pod* and *teardrop* have similar traffic patterns, these two attacks will be grouped and analyzed together in this experiment. The classification performance of the 45 features on these two attacks is listed in Table 4.4.

Table 4.4 The Feature Ranking Table of the *pod* Attack

Rank	Index	Name	Error
1	4	in.ip-frag-rate	0.000118
2	5	in.ip-defrag-error-rate	0.000152
3-45	...	...	0.000608

Because there are 24 pod attack samples and 12 teardrop attack samples in DARPA'98 data, the base-line rate of pod and teardrop attacks is $36/59226 = 0.000608$. Table 4.4 indicates that there are only two features, "in.ip-frag-rate" and "in.ip-defrag-error-rate", which are relevant to detect pod and teardrop attacks.

The Selected Feature Subset

Table 4.5 The Features in Set_{22}

Index	Name	Index	Name
2	in.ip-pkt-rate	14	in.tcp-con-new-aborted
3	in.ip-byt-rate	15	in.tcp-con-half-opened-ratio
4	in.ip-frag-rate	16	in.tcp-con-duration
5	in.ip-defrag-error-rate	17	in.tcp-con-diff-src
6	in.ip-csum-error-rate	18	in.tcp-con-diff-dst
7	in.tcp-pkt-len	19	in.tcp-con-anomalous-entropy
8	in.tcp-pkt-rate	21	in.icmp-pkt-rate
9	in.tcp-syn-pkt-rate	22	in.icmp-byt-rate
11	in.tcp-rst-pkt-rate	23	in.icmp-diff-src
12	in.tcp-con-new-opened	24	in.icmp-diff-dst
13	in.tcp-con-new-closed	45	io.icmp-anomalous-echo-reply

Based on the above feature ranking tables, a feature subset of twenty-two features is selected and listed in Table 5. This feature subset will be noted as Set_{22} from now on, for simplicity.

4.4.3 The Feature Subset of Decision Tree

A decision tree [56] was induced using the DARPA'98 data. Only 8 features were selected by the induced tree (see Table 4.6). It can be seen from the table that the features selected by the decision tree is quite different from the features listed in Table 4.5. For simplicity, this feature subset will be referred to as Set8 in the rest of this chapter.

Table 4.6 The Features Selected by Decision Tree

Index	Name	Index	Name
4	in.ip-frag-rate	17	in.tcp-con-diff-src
12	in.tcp-con-new-opened	22	in.icmp-byt-rate
13	in.tcp-con-new-closed	33	out.tcp-pkt-len
14	in.tcp-con-new-aborted	34	out.tcp-pkt-rate

4.4.4 Comparing Different Feature Subsets

Neural networks are trained using the feature subsets selected in the above experiments. The misclassification rates of these neural networks, after being trained after 2000 epochs, are tabulated in Table 4.7, together with the values of the six objective functions, which are defined in Section 4.2.

Table 4.7 The Performance of Different Feature Sets

s	Set_{45}	Set_{25}	Set_{22}	Set_8	Set_1
$\|s\|$	45	25	22	8	1
Error	0.000540	0.000371	0.000355	0.000321	0.00554
FPR	0.000422	0.000310	0.000327	0.000275	0.000396
FNR	0.00652	0.00372	0.00186	0.00279	0.284
$J_1(s)$	0.000540	0.000371	0.000355	0.000321	0.00554
$J_2(s)$	0.000675	0.000449	0.000427	0.000365	0.00554
$J_3(s)$	0.000675	0.000440	0.000416	0.000353	0.00554
$J_4(s)$	0.000710	0.000514	0.000527	0.000542	0.00268
$J_5(s)$	0.000888	0.000623	0.000634	0.000514	0.00268
$J_6(s)$	0.000888	0.000609	0.000618	0.000497	0.00268

In the table, the "Set$_{45}$" corresponds to the HIDE performance when using all 45 features (see Section 3.5.2). The results show that "Set$_8$" has the lowest misclassification rates and the minimum values in objective functions among the four sets. This proves that the classification accuracy and the computation effectiveness of HIDE can be significantly improved by selecting a proper feature subset.

4.5 Conclusions

This chapter constructed three different feature subsets using different feature filtering methods and evaluated the corresponding classification performance. The experimental results on DARPA'98 data set indicated that both the classification accuracy and the system effectiveness can be significantly improved by selecting a proper feature subset.

Amongst the four feature subsets, the Set$_8$ has the best performance, which includes eight features: *in.ip-frag-rate*, *in.tcp-con-new-opened*, *in.tcp-con-new-closed*, *in.tcp-new-aborted*, *in.tcp-con-diff-src*, *in.icmp-byt-rate*, *out.tcp-pkt-len* and *out.tcp-pkt-rate.*

By using the eight features in Set$_8$, HIDE can effectively reduce the values of the four objective functions by almost a half. This proves the importance for an IDS to properly select the feature set it monitors to maximize the classification accuracy and to minimize the computation cost.

However, please note that this feature set may not be the optimum feature set, in terms of the misclassification rate and the number of features, for HIDE to detect DoS attacks. As reported by John [57], the process of feature subset selection should depend not only on the features and the objective functions, but also on the learning algorithm. In the future, an experiment using wrapper searching methods might be carried out to search the optimum feature subset for backpropagation networks.

Moreover, it needs to be kept in mind that the whole feature identification process is data driven, in that the selected feature subsets are highly dependent on the training data. Therefore, special care is needed to avoid over-fitting the feature subset for a specific training data.

Given the constant fluctuation and evolution of both background and attack traffic in Internet, the real objective of feature selection is not to find a feature subset, which may be optimal for a specific data set but generalize poorly on other data files, but to select a feature subset that is capable of operating at optimal or near-optimal conditions over a broad range of network settings. Therefore, the next step for this research is to apply this feature selection process to data collected from various sources to further understand the correlations between HIDE features and DoS attacks and to extensively verify the selected feature subsets.

CHAPTER 5

OPTIMIZING HIDE

The results in Section 3.5 indicated that HIDE can effectively detect *DoS* attacks using PDF statistics and neural classifiers with very high accuracy. This section reports a series of research efforts to optimize the performance of HIDE.

The study of various PDF partitioning algorithms is reported in Section 5.1. Section 5.2 explores the feasibility of reducing the computation and storage complexity of HIDE using Wavelet compression. Section 5.3 compares the performance of different similarity metrics. Section 5.4 tests the classification performance of different neural networks.

5.1 The Study of PDF Partitioning Algorithms

Representing network measurements into PDF formats involves partitioning the sample spaces of the measurements into a set of complete and non-overlapping bins and calculating the frequencies of the observed events that fall within the bins. Traditional PDF algorithms, e.g. NIDES, built PDF histograms using equally sized bins, while the total number of bins utilized is determined by expediency. For example, NIDES used 32 bins, chosen to be large enough so details will not be lost at the cost of additional computational complexity. However, the choice of the partitioning structure may have significant consequences in the accuracy and the effectiveness of the numerical representation of a monitored parameter. Yet, little systematic investigation seems to have been undertaken on this issue. Here three different partitioning schemes, uniform, logarithmic and percentile, have been tested in [58].

Uniform Partitioning Scheme

The *uniform* partitioning algorithm divides the sample space into bins of equal parameter width. Assume $x_0, x_1, \ldots, x_N$ is a partition of the sample space with x_0 the minimum and x_N the maximum, where N is the total number of bins. The partition can be calculated using Equation 5.1.

$$\begin{aligned} x_0 &< x_1 < x_2 < \ldots < x_N \\ x_1 - x_0 &= x_2 - x_1 = \ldots = x_N - x_{N-1} \\ x_i &= x_0 + i \times \frac{x_N - x_0}{N} \end{aligned} \tag{5.1}$$

Logarithmic Partitioning Scheme

In the *logarithmic* partitioning scheme, the sample space is segmented using Equation 5.2.

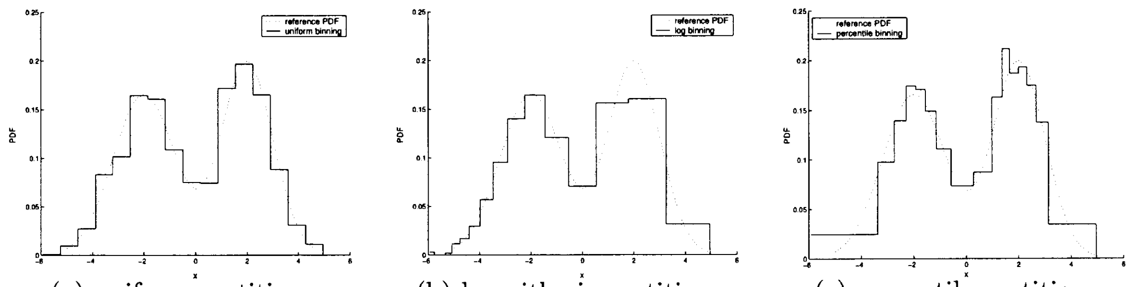
$$\begin{aligned} \frac{i}{N} &= \log_{(1+x_N-x_0)}(1 + x_i - x_0) \\ x_i &= x_0 - 1 + (1 + x_N - x_0)^{\frac{i}{N}} \end{aligned} \tag{5.2}$$

Percentile Partitioning Scheme

Assume that $F(x)$ is the cumulative distribution function (CDF) of a PDF. The *percentile* partitioning algorithm generates bins so that all bins have equal probability value (or mass) following Equation 5.3.

$$\begin{aligned} F(x_0) &= 0 \qquad F(x_N) = 1 \\ F(x_1) - F(x_0) &= \ldots = F(x_N) - F(x_{N-1}) \\ F(x_i) &= F^{-1}(\tfrac{i}{N}) \end{aligned} \tag{5.3}$$

Sample PDFs using these three partitioning schemes are illustrated in Figure 5.1. It can be seen that the *log* partition has small widths at the lower end and large widths at the higher end. The *percentile* partition uses small bins to represent the middle part of the distribution but uses very large bins to represent the two tails. The *uniform* partition gives equal emphasis to all bins.

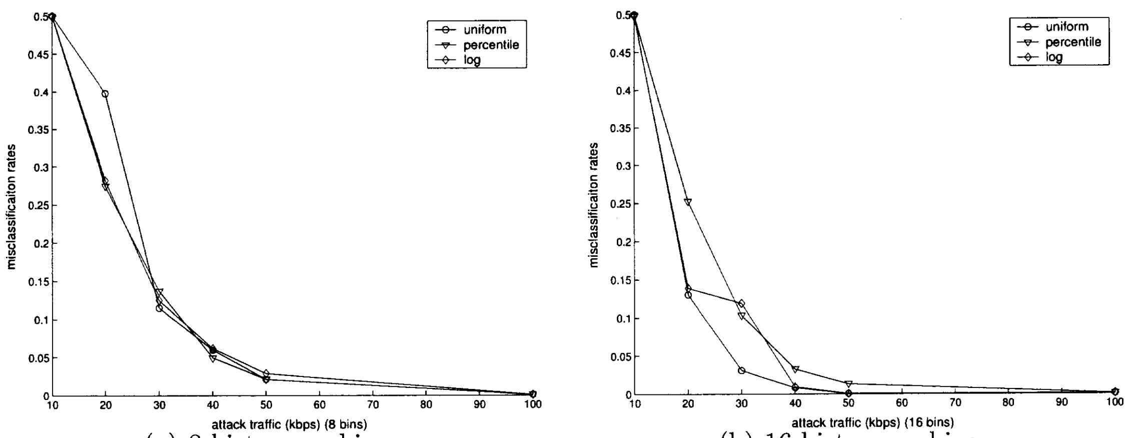


(a) uniform partition (b) logarithmic partition (c) percentile partition

Figure 5.1 Sample PDF figures with different partitioning algorithms.

The misclassification rates of 8-bin and 16-bin PDF representations as functions of attack level are plotted in Figure 5.2. From the figure, the readers can see that, as the attack level increases, the misclassification probabilities of the systems decrease, as might be reasonably expected. When the attack levels are low (say 40 kbps), it is difficult for the PDF schemes with fewer bins (e.g. 4 bins) to detect the attacks, while the PDFs with more bins (e.g. 16 bins) still result in low misclassification rates. This supports the theory that using PDFs to represent system variables improves the classification accuracy since they abstract more detailed information that might be useful for intrusion detection systems.

As the total numbers of bins increase, the behaviors of the three partitioning algorithms become more similar. At 64, they all perform about the same. This can be explained by observing that the differences between the real PDFs of the monitored parameters and their histogram representations decrease as more bins are used to represent them. As the total numbers of bins changes from 4 bins to 64 bins, in all sub-figures, the misclassification rates at first decrease, denoting improvement,



(a) 8 histogram bins. (b) 16 histogram bins.

Figure 5.2 Misclassification rates vs. attack levels.

however, when the total number of bins rises beyond 16, the curves become almost flat, denoting no further performance improvement. This observation suggests that PDF representations with 16 bins could be the optimum choice for the trade off between processing complexity and system performance. Please refer to [58] for more detailed results.

5.2 Compressing PDFs Using Wavelet Compression

Building and handling PDFs consumes much in system processing power, memory and storage resources. Therefore, an effective data compression algorithm has the potential to significantly enhance the system efficiency. This section reports the experiments on compressing PDF data using Wavelet compression.

Among the many wavelet families that exist, [65–68], a wavelet is selected from each of the following families: the classical Haar basis (called *haar* afterward), the Symlets basis (called *sym2* afterward), Coiflets (called *coif3* afterward), and Daubechies (called *db4* afterward) wavelets. Samples of compressed PDFs using one of these four wavelet compression algorithms are plotted in Figure 5.3.

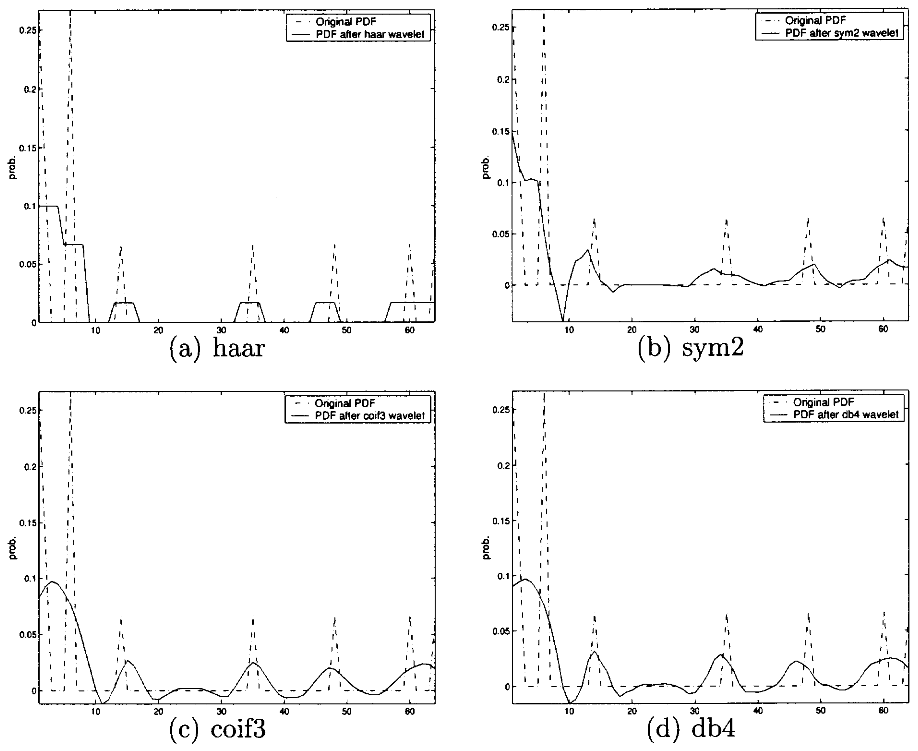


Figure 5.3 The sample figure of a PDF with different wavelet compressions.

Table 5.1 Number of Wavelet Coefficients per Range

Wavelet	Number of Coefficients				
	Range 1	Range 2	Range 3	Range 4	Range 5
haar	64	32	16	8	4
sym2	64	33	18	10	6
coif3	64	35	21	14	10
db4	64	40	28	22	19

Experiments with various wavelet compression ranges (see Table 5.1 have been carried out and reported in [69]. In the table, PDFs at compression range 1 correspond to uncompressed PDFs. Note that the wavelets *coif3* and *db4* require the largest number of coefficients at each range, while the wavelets *haar* and *sym2* provide the greatest amount of compression.

The above-mentioned wavelet algorithms were used to compress the PDF data files collected from the OPNET simulations (see Section 3.5.1), in which the PDFs are represented using *unfiorm* partitioning algorithm, see Section 5.1, with 64 bins into sets of wavelet coefficient. To compare the system performances under various wavelet compression ranges, the compressed PDFs were reconstructed from the wavelet coefficients found and then processed by the statistical modules and the neural network classifier.

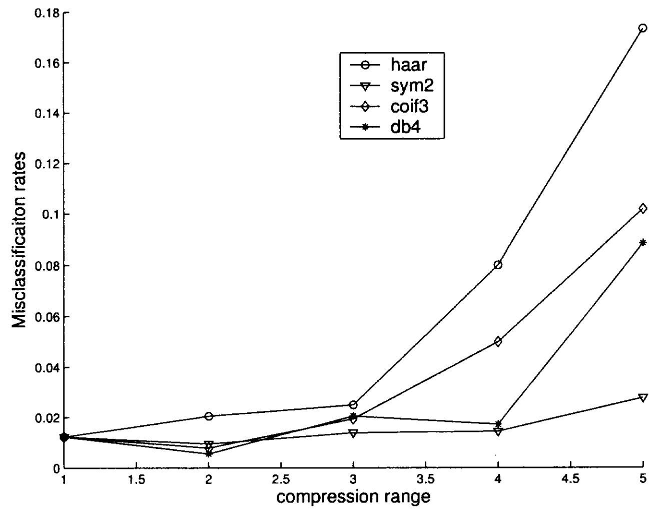


Figure 5.4 The misclassification rates of HIDE at different wavelet compression ranges.

The misclassification rates of HIDE, when trained using these reconstructed PDFs, are plotted in Figure 5.4, in which the x-axis represents the five compression ranges and the y-axis represents the misclassification rates. It can be seen from the figure that the curve of wavelet compression algorithm sym2 rises slowly, and, even for compression range 5, the algorithm still shows strong performance with misclassification rate about 2%. For the other three wavelet bases, the system performance is satisfactorily for ranges 1 to 3, but then deteriorates for ranges 4 and 5.

The experimental results indicated that the wavelet compression algorithm *sym2* is a more appropriate choice for PDF compression. In practice, it was found that *sym2* wavelet compression with compression range of 3 (18 wavelet coefficients) is a safe choice for HIDE to maintain a satisfying performance while boosting system resource efficiency by almost four fold.

5.3 The Effectiveness of Similarity Metrics

This section investigates the effectiveness of different similarity metrics in intrusion detection. The four metrics studied are: χ^2 test (see Equation 3.1); KS test (see Equation 3.2); AKS test (see Equation 3.3); and a simplified single-number statistic (so called SG test, see Equation 5.4). The SG test is used to simulate intrusion detection using traditional rate-based single-number parameters.

$$D = \frac{p(r)}{p_{max}} \tag{5.4}$$

where r is the average value of the monitored parameter within a time window; $p(r)$ is the probability of r in the reference distribution; and p_{max} is the maximum probability of the reference distribution.

The classification performance of these four metrics, when detecting the OPNET data set, is depicted in Figure 5.5. The SG metric performs the worst, especially when attack intensities range from low to medium. This indicates that, by using the PDF statistics, IDSs can potentially improve the detection rate of early detection of *DoS* attacks.

The AKS metric has slightly better performance than the KS statistic. The χ^2 test has the lowest misclassification rates when the attack intensity is low. The

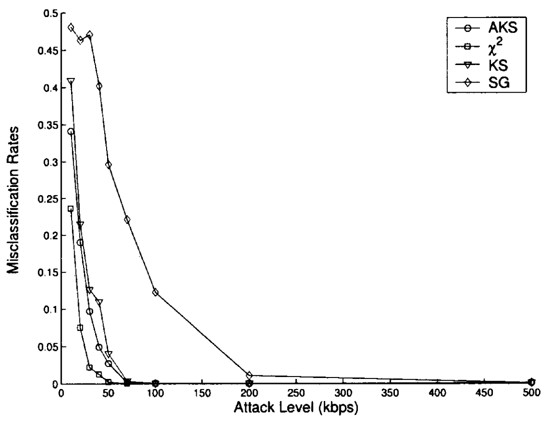


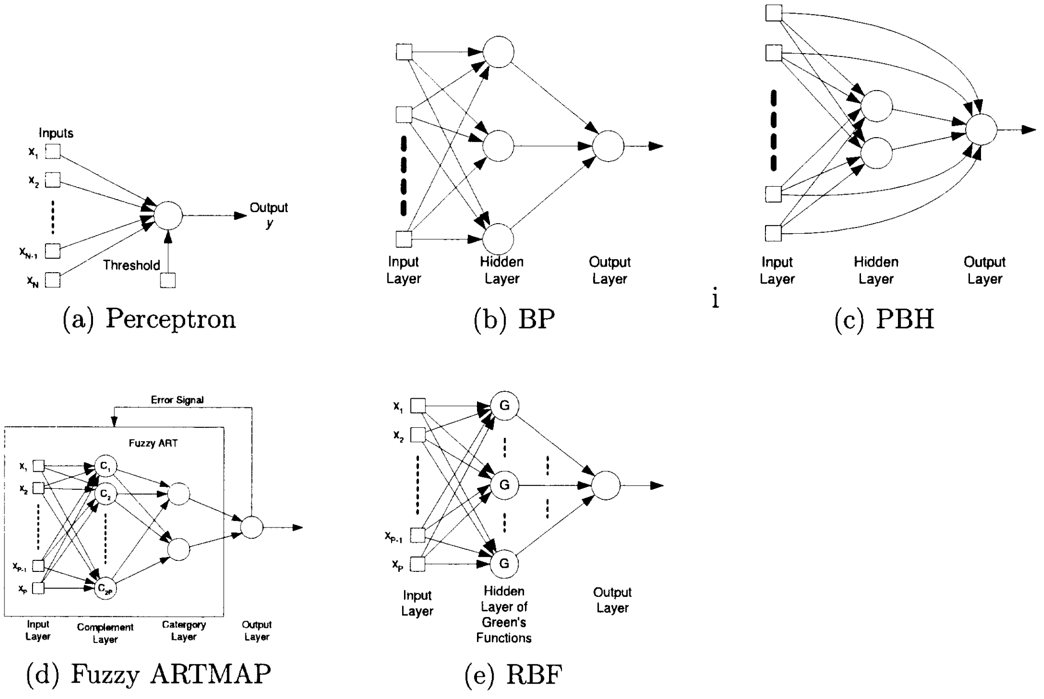
Figure 5.5 The misclassification rates of different similarity metrics.

plausible explanation for this observation is that, because the underlying driving distributions of the OPNET simulations are "Poison" and "Exponential", which are memoryless and individually independent, the resulted χ^2 distances closely follow their theoretical distributions. Therefore, the χ^2 test can accurately tell the difference between normal and attack PDFs. However, in real network environments, where the traffic distributions are often bursty and self-similar [59], the above "Poisson" and "Exponential" assumptions are no longer valid. Due to the inherent distribution sensitiveness, χ^2 test may not be a good choice for real network detection. Therefore, the AKS test is chosen as the similarity metric for most of this work.

5.4 The Study of Neural Network Classifiers

Neural networks are widely considered as an efficient approach to adaptively classify patterns, but high computation intensity and long training cycles have hindered their applications in real-time systems. In [7, 19], BP neural networks were used to detect anomalous user activities. In [39], a hybrid neural network paradigm, called perceptron-backpropagation-hybrid (or PBH) network [60], which is a superposition of a perceptron and a small backpropagation network, was deployed to detect intrusions.

As the core of decision making modules of many anomaly IDSs, classification techniques have profound impact on system performance and efficiency, but little research has been carried out to compare the effectiveness of different neural network classifiers as applied to ID problems. In order to comprehensively investigate the performance of neural networks, five different types of neural networks are examined in this section: Perceptron, BP, PBH, Fuzzy ART MAP and RBF.



(a) Perceptron (b) BP (c) PBH (d) Fuzzy ARTMAP (e) RBF

Figure 5.6 The models of neural classifiers tested.

The perceptron [61], Figure 5.6(a), is the simplest form of a neural network used for the classification of linearly separable patterns. It consists of a single neuron with adjustable synapses and threshold. Although the data sets will not, in general, be linearly separable, the perceptron was used as a baseline to measure the performance of other neural networks.

The Backpropagation network [61], Figure 5.6(b), or BP, is a multi-layer feed-forward network, which contains an input layer, one or more hidden layers, and an output layer. BPs have strong generalization capabilities and have been applied successfully to solve some difficult and diverse problems. The classification performance of BP networks, with the number of hidden neurons ranging from 2 to 8, was tested.

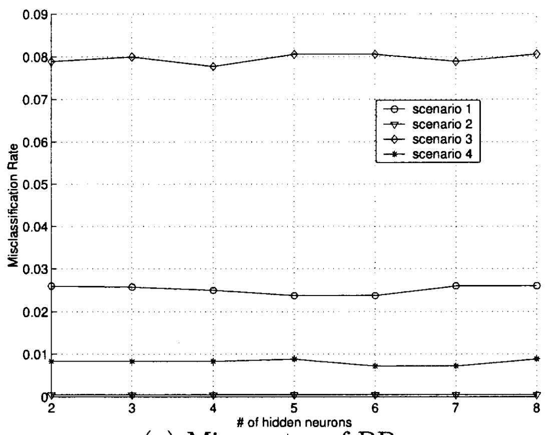
Perceptron-backpropagation hybrid network [60], or PBH, Figure 5.6(c), is a superposition of a perceptron and a small backpropagation network. PBH networks are capable of exploring both linear and nonlinear correlations between the input stimulus vectors and the output values. The classification performance of PBH networks, with the number of hidden neurons ranging from 1 to 8, was tested.

The Fuzzy ARTMAP, [62], in its most general form is a system of two Fuzzy ART networks ARTa and ARTb whose F2 layers are connected by a subsystem referred to as a "match tracking system". A simplified version of Fuzzy ARTMAP [63], Figure 5.6(d), which is implemented for classification problems, was used in this experiment. The ARTMAP networks, with the number of category neurons ranging from 2 to 8, was tested.

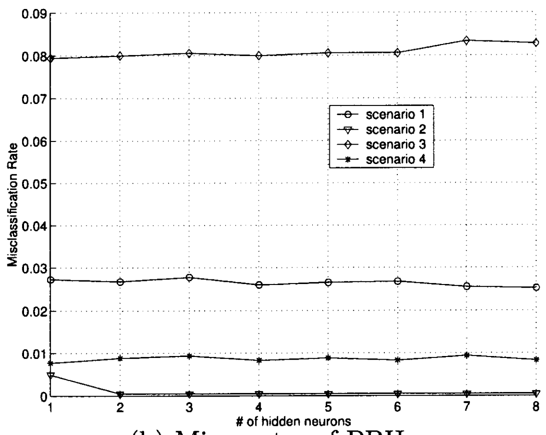
Radial-basis function network [61], or RBF, Figure 5.6(e), involves three entirely different layers. The input layer is made up of source nodes. The second layer is a hidden layer of high enough dimension, which serves a different purpose from that in a BP network. The output layer supplies the response of the network to the activation patterns applied to the input layer. The RBF networks, with hidden neurons ranging from 2 to 8, was tested.

The misclassification rates of BP and PBH neural networks are plotted in Figure 5.7. The detailed results of all these four classifiers can be found in the paper [64]. Due to the space limitations, they are not included in this section.

The conclusion in [64] was that BP and PBH networks had similar detection performance, and that the both neural networks consistently performed better than the other three types. The misclassification rates of these two networks do not



(a) Misc. rates of BP.



(b) Misc. rates of PBH.

Figure 5.7 The results of BP and PBH classifiers.

decrease as the number of hidden neurons increases. The explanation could be that the sample spaces of attack and normal patterns are clearly divided after being processed using PDF statistics. They are not challenging enough for BP and PBH. Therefore, the BP network is chosen as the HIDE's classifier in most of this work.

CHAPTER 6

METHODS OF CLASSIFIER TRAINING IN A PRODUCTION NETWORK USING TEST NETWORK INFORMATION

6.1 The Challenge in Training Classifiers for a Production Network

In a production network, during its intended operation, an anomaly intrusion detection system is expected to find anomalous activities. As shown in Figure 2.2, the pattern classifier processes the current activity pattern and rates it with a similarity score as to whether it is more similar to normal or anomalous activity. Similar to most other learning techniques using supervised training, the HIDE classifier, which is a back-propagation neural network, needs to be trained with data records of the typical (normal) and attack-labeled (anomalous) variety in sufficiently large amounts. Moreover, if one desires to distinguish between different classes of attacks or perhaps individual attacks, data for all such different attacks need to be made available for training. Such data will enable the neural net classifier to learn the difference between normal and anomalous activities, thus achieving high detection and low false alarm rates. In a test or laboratory network, the researchers and developers can investigate and record the launching of all desired known attacks, selecting from a library of attacks, with accurate labels. Such investigations of attacks can be carried out and recorded for various background traffic types and levels.

The challenge that HIDE faces is that, while both normal and attack data are easily and conveniently available for a test network, only normal data are routinely available for a production network. Hence, the classifier of the IDS will probably be ill-trained for the new network environment and thus unsuitable for use without changes if (1) the new network is sufficiently different from the test network, so the new background will not be close to that of the test network, or (2) the IDS lacks training that includes attack data for the new network. One or the other, or more likely both conditions hold in most realistic cases of installations of a new intrusion detection system and thus the IDS classifier will need to be re-trained at some stage during or soon after installation.

An anomaly intrusion detection tool that is to be installed in a new real network environment can reasonably expect some initialization period that will provide sufficient estimates of the normal behavior of the network. In a typical setting, by far most of the traffic that the IDS sees is normal with only a sprinkling of attack traffic. Thus, algorithms could conceivably be used to separate out the normal traffic from any attacks, whenever they might occur in the new network, based on intensity comparison considerations alone. Such algorithms, employing clustering techniques, have been investigated elsewhere [29].

Additionally, it may be that, during the initialization phase, all the traffic in the new setting is in fact normal. Even if some attack traffic could be found in the new network, it may be difficult, at this stage, to characterize what type of attack it is. Therefore only normal traffic models in the new network setting can be observed, but no clear description of network attacks can be expected. Although theoretically possible, it would be unreasonable and unrealistic to launch attacks in the network to be protected, solely for the benefit of the learning phase of the IDS. Most likely it would not be permitted to take place. Thus, somehow, while using the normal and attack data from a test network as well as the normal data from the new network, it needs to be ensured that the classifier is properly and correctly trained.

Two alternative solution approaches to this challenge, the *re-use classifier* and the *grafted classifier*, were proposed in [70]:

- *The re-use classifier* method is straightforward and consists of employing the test-network classifier in the new network, **as is** after being trained only in the test network, which is without any modifications at the beginning. This approach is expected to perform adequately well if the new network is similar in architecture and usage to the test network, but will likely increasingly deteriorate in performance the more dissimilar the two settings are.

- *The grafted classifier* method essentially consists of abstracting a database of attacks, in their "pure" forms, from the test network and then "**grafting**" these attacks onto the normal traffic in the new network, thus "transplanting" the attack models from the test network to the production network.

In this way, the grafted classifier method will provide adequate attack samples as well as normal samples in the new network setting, thus enabling the training of the classifier in the new network. Experiments have been performed to investigate the effectiveness of these two techniques in an experimental setting, where both have been found to be effective. The grafted classifier approach will work adequately for the class of parameters where the effect of the attacks is of "additive" character to that of the background, which is the case for the *DoS* attacks that HIDE is designed to detect.

Nevertheless, it should be noted that the expectation regarding these techniques is *not that they generate the final best model of an attack, but that they will provide a good "seed" of such a model*, allowing the system to adapt the model by "learning" from then on, thus bootstrapping onto an accurate attack model from an initially only adequate estimate. The HIDE employs an adapting algorithm that adjusts the normal as well as the attack models, by using the descriptions of the current normal and attack data, after they are detected to be so.

The grafted classifier approach described above may, in general, depend on the nature of the monitored parameter as well as the type of attack. Basically, it relies on the assumption that the normal background models vary most from network to network while the attack models vary least. This assumption is intuitively reasonable and has been observed to be the case for many types of networks and attacks. The normal background network traffic changes in time and from network to network. On the other hand, the behavioral profiles of various kinds of attacks are comparatively stationary and exhibit similarities from network to network.

In more detail, this approach involves extracting attack profiles, for each monitored parameter, from a test network and then applying them to a new production network. Firstly, the samples of attacks and background traffic are extracted from the data collected in a test network. Secondly, the attack instances are calculated by removing the components of normal traffic from the attack models. Thirdly, the attack samples for the new network can be simulated by combining and merging the normal samples of the production network with the attack instances that have been extracted from the test network. Finally, the IDS classifier can be trained using the attack data simulated in the previous step together with the normal data samples.

The definitions of *attack samples*, *normal samples*, *attack instances* and *attack profile*, for each parameter, are provided as below:

- *Attack Hybrid* is the observed traffic pattern, which is represented in PDF format, with both normal and attack traffic.
- *Normal Sample* is the observed traffic pattern with normal traffic only.
- *Attack Abstract* is the instance of attack traffic, in its "pure" form, by removing the normal traffic components from attack samples.
- *Attack Profile* is an aggregated database of all attack instances.

6.2 Attack Modeling and Simulation

The models of normal network traffic are generally bursty and vary considerably in a network from time to time and from network to network, therefore it is hard to predict or estimate the models of normal traffic in advance. However, because the schemes and patterns of particular attacks often have their own specific characteristics, their effects on the normal traffic may be fairly consistent. Thus, it may be feasible to estimate attack models based on the knowledge collected from a test network, at least well enough to serve as seeds in the new network setting. The flow diagram of the attack modeling and simulation is shown in Figure 6.1, which involves the following steps:

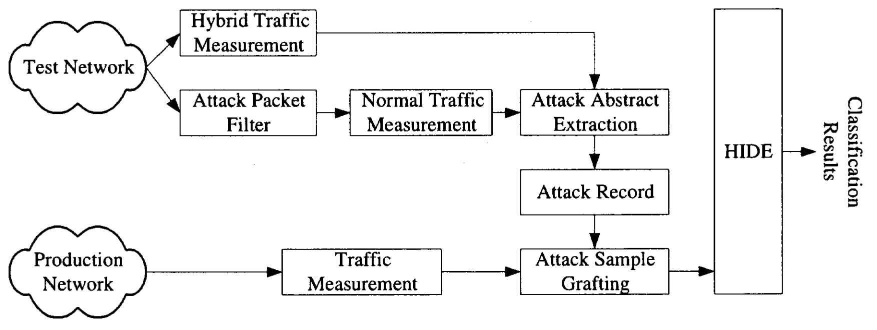


Figure 6.1 The phases of attack estimation.

1. **Hybrid Traffic Sampling in the Test Network**: measures the network traffic, including both normal and attack packets, of the test network, into PDF data samples.

2. **Attack Packet Filter**: blocks the attack packets so that only normal packets can pass through this block.

3. **Normal Traffic Sampling in the Test Network**: measures the network traffic, including normal packets only (the attack packets are filtered out by the "Attack Packet Filter"), of the test network, into PDF data samples.

4. **Attack Abstract Extraction**: extracts the attack instances by removing the normal traffic components from the attack samples.

5. **Attack Record**: is the database of all attack abstract instances extracted from the previous step.

6. **Normal Traffic Sampling in the Target Network**: measures the normal traffic patterns in the target network.

7. **Attack Transplant Sample Grafting**: at this stage both the attack abstract instances, which are collected from the test network, and the normal samples, which are collected from the target network, have all become available. However, the real objective of these data is to simulate enough attack samples to be used as training, test and validation data. Explicitly, this algorithm needs to be able to randomly generate the presumed attack samples that are PDFs that statistically predict the traffic patterns, should the same attack be observed in the target network. Subsequently, these simulated attack samples together with the normal samples of the target network are both used to train the classifier of HIDE.

6.2.1 Attack Abstract Extraction

Assume that AH is a measured attack sample, which contains both attack and normal traffic. BN is the measure normal sample, which contains the normal traffic only. Then the equations to calculate the attack instance are given in Equation 6.1

$$\begin{aligned} \mathrm{AA}(i) &= \frac{\mathcal{N}_{\mathrm{AH}_t} * \mathrm{AH}_t(i) - \mathcal{N}_{\mathrm{BN}_t} * \mathrm{BN}_t(i)}{\mathcal{N}_{\mathrm{AH}_t} - \mathcal{N}_{\mathrm{BN}_t}} \\ \mathcal{N}_{\mathrm{AA}} &= \mathcal{N}_{\mathrm{AH}_t} - \mathcal{N}_{\mathrm{BN}_t} \end{aligned} \tag{6.1}$$

Where:

- i: i^{th} bin of a PDF sample.
- AH_t: is the measured attack traffic sample in the test network.
- BN_t: is the measured normal traffic sample in the test network.
- $\mathcal{N}_{\mathrm{AH}_t}$: the number of samples of AH_t.
- $\mathcal{N}_{\mathrm{BN}_t}$: the number of samples of BN_t.
- AA: is the calculated attack abstract instance.
- $\mathcal{N}_{\mathrm{AA}}$: is the number of samples of AA.

6.2.2 Attack Transplant Sample Grafting

The attack sample grafting is the reversed process of the attack instance extraction step. First, an attack abstract, AA, is randomly chosen from the Attack Record; then, the attack transplant sample, AT_g, is generated by grafting the selected attack abstract onto a normal sample collected from the target network. The equation of attack transplant grafting is given in Equation 6.2.

$$\begin{aligned} \mathrm{AT}_g(i) &= \frac{\mathcal{N}_{\mathrm{AA}} * \mathrm{AA}(i) + \mathcal{N}_{\mathrm{BN}_p} * \mathrm{BN}_p(i)}{\mathcal{N}_{\mathrm{AA}} + \mathcal{N}_{\mathrm{BN}_p}} \\ \mathcal{N}_{\mathrm{AT}_g} &= \mathcal{N}_{\mathrm{AA}} + \mathcal{N}_{\mathrm{BN}_p} \end{aligned} \tag{6.2}$$

Where:

- i: i^{th} bin of a PDF sample.
- AA: is the calculated attack abstract instance.
- N_{AA}: is the number of samples of AA.
- BN_p: the normal traffic sample of the target production network.
- N_{BN_p}: the number of samples of BN_p.

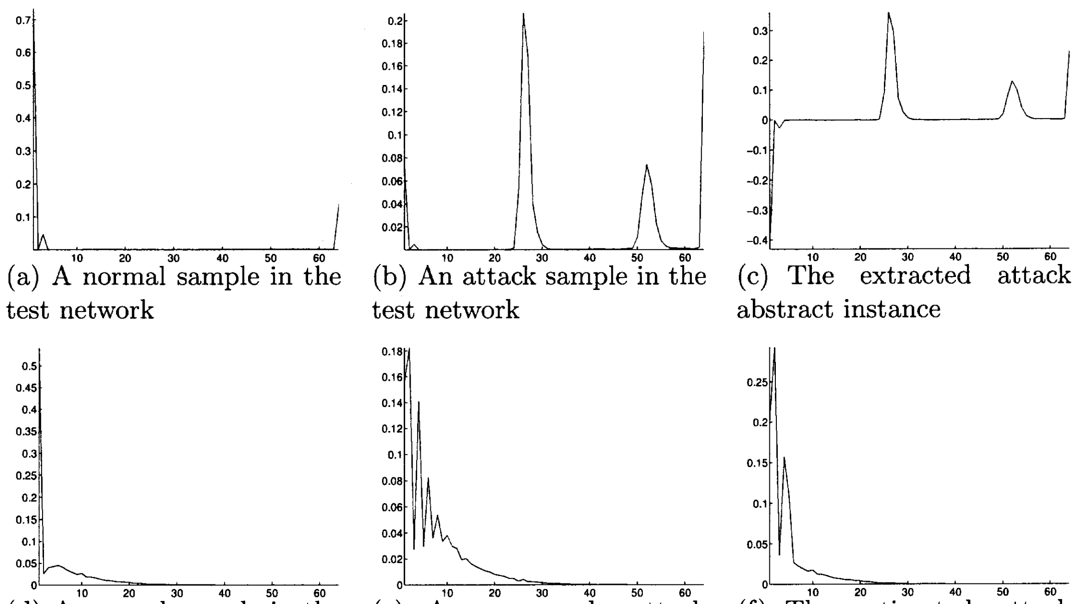
Please refer to [70] for the detailed information about the algorithms and the pseudo codes of attack instance extraction and simulation.

6.2.3 PDF Figures of a Grafted Attack Sample

Example PDFs of the measured normal and the measured attack samples, the extracted attack abstracts and the resulted attack transplant samples are plotted in Figure 6.2 for some parameter. Figure 6.2(a) and Figure 6.2(b) are the measured normal and attack PDF samples in a test network. Figure 6.2(c) is the extracted attack abstract instance based on Figure 6.2(a) and Figure 6.2(b). Figure 6.2(d) and Figure 6.2(e) are the measured normal and attack samples in a target network. Figure 6.2(f) is the simulated attack transplant sample in the target network. From the figures, it can be seen that, although the normal traffic samples in the two networks are quite different (as seen by comparing Figure 6.2(a) and Figure 6.2(d)), the simulated attack transplant sample is visually close to the measured attack sample in the target network (as seen by comparing Figure 6.2(f) and Figure 6.2(e)).

6.3 The OPNET Experimental Results

The first experiment is designed to evaluate the performance of the "graft" and "re-use" algorithms, when both the test and the target networks have similar network settings. The OPNET simulation data sets (see Section 3.5.1) are used in this experiment.



(a) A normal sample in the test network

(b) An attack sample in the test network

(c) The extracted attack abstract instance

(d) A normal sample in the target network

(e) A measured attack sample in the target network

(f) The estimated attack transplant sample in the target network

Figure 6.2 The PDF histograms corresponding to the steps utilized in estimating an attack transplant sample of a parameter.

The 600kbps network (see Table 3.1) serves as the test network (A). The background and attack traffic data of the test network are used for training the test classifier. This test classifier is used in the investigation of the efficacy of the *re-use* classifier method. These data are also used to calculate the attack abstract instances by using the algorithms described earlier that are part of the grafted classifier method.

The 2Mbps network (see Table 3.1) serves as the production network (B). The background traffic data of the network B are used to compute the background models and then to estimate the attack models in the production network. The attack models then generate simulated attack transplant samples for training.

6.3.1 Model Similarities

To further quantify the similarities between the measured and the estimated models, three different similarity measurements are calculated using the similarity equation listed in Equation 3.3: (1) the similarity between the actual background model and the actual measured attack model (called $\mathrm{A_aB_a}$ from now on)(here the term "actual" refers to measured, rather than estimated, quantities in the simulated network, not live network data); (2) the similarity between the actual background model and the estimated attack model (called $\mathrm{A_eB_a}$ from now on); (3) the similarity between the actual attack model and the estimated attack model (called $\mathrm{A_eA_a}$ from now on); (4) the similarity between the actual 600kbps-background model and the actual 2Mbps-background model (called $\mathrm{B}_{a_{600k}}B_{a_{2M}}$ from now on). The average of the similarity results are tabulated in Table 6.1.

Table 6.1 The Average Similarity Results

attack	Average Similarities			
(kbps)	$\mathrm{A_aB_a}$	$\mathrm{A_eB_a}$	$\mathrm{A_eA_a}$	$\mathrm{B}_{a_{600k}}B_{a_{2M}}$
10	0.958	0.982	0.962	0.565
50	0.904	0.927	0.941	0.603
100	0.837	0.84	0.896	0.555
200	0.706	0.677	0.839	0.599

In examining Table 6.1, it can be noticed that, when the attack level is very low compared with the background, i.e., 10 to 50 kbps, all models are close to each other, and in fact, none of the classifiers can reliably detect attacks. At higher attack levels, i.e., 100-200 kbps and more, the estimated attack models remain very similar to the measured attack models but different from the actual normal models. Therefore, these results suggest that the attack models can be estimated with adequate correctness in that they are quite similar to the actual models. The last column strongly indicates that the background at 600kbps and 2Mbps network conditions are consistently very different from each other.

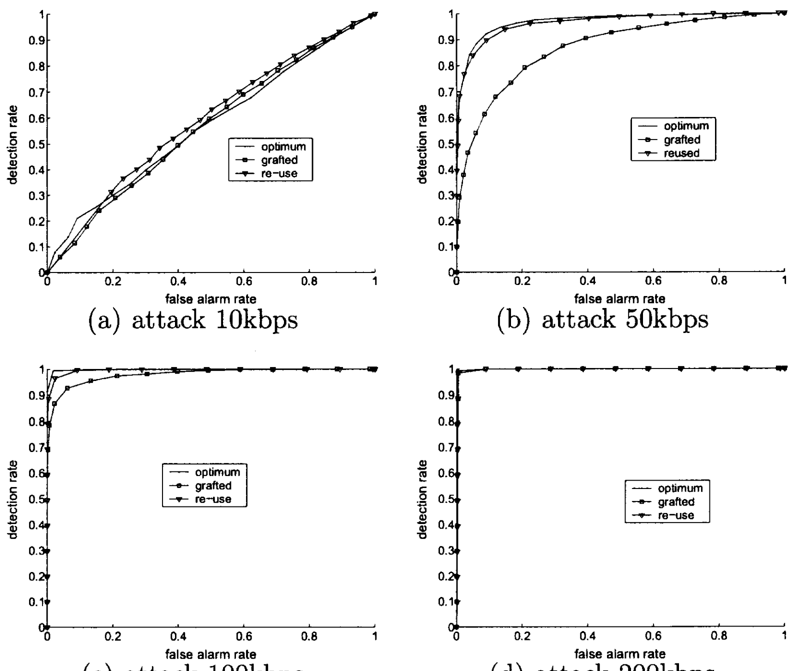
6.3.2 Comparison of Classification Performance

Three classification experiments are performed to examine the classification performance outcomes of HIDE for the networks at hand for the two approaches, the *re-use* and the *grafted* methods.

1. The actual PDF data of the production network B are used for the training, testing and validation of the HIDE classifier. The results are used as the baseline of the system performance in that they should be the best that can be achieved for the actual 2 Mbps network. For simplicity, this experiment is called the *optimum* for experimental outcomes.
2. The generated PDF data, using the grafted classifier method, are used for training the HIDE classifier, while the actual data are used for testing. The results reflect the performance of the "grafted" method and are referred to as the *grafted* method experiment.
3. The 600 kbps PDF data are used for both training and testing of the HIDE classifier. Subsequently, the classifier is used to validate the PDF data from the production network. Thus, this corresponds to examining the efficacy of the "re-use" method and is to be referred to as the *re-use* method experiment.

The ROC (Receiver Operating Characteristic) curves of the three experiments under different attack levels are plotted in Figure 6.3.

From the figures, it can be seen that the classification performance of HIDE improve, as the attack intensity gets higher, as expected. Also, the classification performance of the *optimum* measurements is best. As mentioned earlier, at the very low attack levels, all classifiers are basically failing and any measured classification results bear little meaning or interest; that is the case here for the 10 kbps graphs. At higher, more meaningful, attack rates, the classification performance of the two methods, the *grafted* and *re-use* classifiers, are quite encouraging and basically on par with each other and only a little inferior to the *optimum* results.



(a) attack 10kbps (b) attack 50kbps

(c) attack 100kbps (d) attack 200kbps

Figure 6.3 The ROC curves at different attack levels.

Table 6.2 Misclassification Rates after Training for 300 Epochs

attack	Misclassification Rates		
(kbps)	*optimum*	*grafted*	*re-use*
10	0.432	0.446	0.424
50	0.0783	0.204	0.0944
100	0.00778	0.0633	0.0267
200	0	0.00278	0.00556

The misclassification rates of HIDE for the *optimum*, *grafted* classifier and *re-use* classifier experiments are listed in Table 6.2. The values seen in the table follow what was observed from Figure 6.3. The misclassification rates of the *optimum* experimental measurements are better but close to those found for the *grafted* and *re-use* classifier experiments. This indicates that the latter two modeling and estimation algorithms can be used as effective starting points, when migrating an anomaly IDS from the test network to an unfamiliar network setting, where accurate attack models are not available. However, the *grafted* approach performs better as the environment becomes increasingly unfamiliar.

6.4 The DARPA-CONEX TESTBED Experimental Results

The second experiment is designed to evaluate the performance of these graft/reuse approaches in detecting *DoS* attacks collected from two networks with disparate network topologies and traffic models.

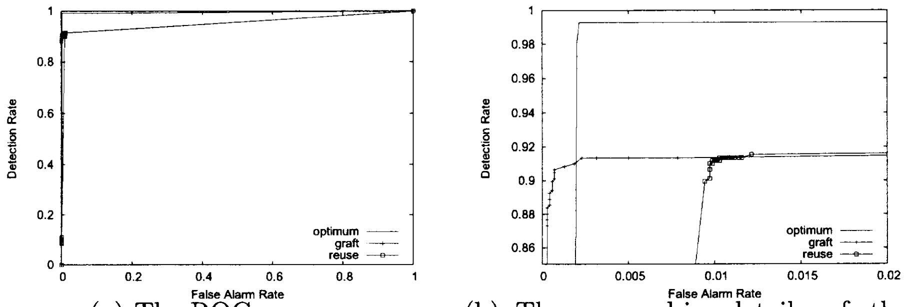
The DARPA IDEVAL network (see Figure 3.6) serves as the test network (A). The background and the *DoS* attack traffic data are used as the training data to train the "re-use" classifier. These data are also used to extract the attack profile in the same way as the previous subsection.

The CONEX TESTBED network (see Figure 3.7) is used as the production network (B). The background traffic data, together with the attack profile of the DARPA IDEVAL network, are used to simulate the grafted attack samples, which in turn are used to train the "graft" classifier.

Table 6.3 The Misclassification Rates on the CONEX TESTBED Data

Experiment	ERR	FPR	FNR
optimum	0.00290	0.00257	0.00704
grafted	0.00791	0.000570	0.0986
re-use	0.0158	0.00984	0.0898

In the same fashion as what has been done in Section 6.3.2, the three classifiers, *optimum*, *grafted* and *re-use*, are trained to detect the *DoS* attacks in the CONEX TESTBED. The misclassification rates are listed in Table 6.3. As expected, the *optimum* classifier has the lowest misclassification rate, as low as 0.290%. The *grafted* classifier comes in second with a 0.791% error rate. The *re-use* classifier performs the worst with a 1.58% misclassification rate.



(a) The ROC curves

(b) The zoomed-in details of the upper left corner

Figure 6.4 The ROC curves of the CONEX TESTBED experiment.

The ROC curves of these three classifiers are plotted in Figure 6.4. Figure 6.4(a) shows the ROC curves in the regular scales and Figure 6.4(b) gives the details at the upper left corner of Figure 6.4(a), which is the region of interests. The figures show that the detection (or true positive) rate of *optimum* classifier is about eight percent higher than the other two classifiers. Interestingly, the *grafted* approach has a slightly lower false positive rate than the *optimum*, which is highly desirable since high false positive rates are the major drawback of intrusion detection systems.

The above observations show that the "graft" approach has a performance advantage over the "reuse" approach in the terms of false positive and misclassification rates, when both are used as bootstrapping classifiers of HIDE for the CONEX TESTBED network.

6.5 Conclusion

The classifier of an IDS, which is well-trained in a test network, could be ill-trained in an unfamiliar network setting. This is because, while typical background traffic data are available in the new network, attack data usually is not. To solve this problem, two techniques, the *re-use* classifier and the *grafted* classifier methods, are proposed and presented in this chapter to bootstrap the classifiers of a production network by using the attack information obtained from a test network.

Two experiments have been carried out by using HIDE, an intrusion detection prototype, to compare the *optimum* outcomes (when the classifier knows the attacks in the unfamiliar network) to the results for the *grafted* and *re-use* classification methods: the first experiment used data sets generated by OPNET simulations at various background and attack intensity settings; the second experiment used TCPDUMP trace files derived from the DARPA intrusion detection evaluation projects and the CONEX TESTBED network of the CONEX lab in NJIT.

It has been seen from the results of the first experiment (see Section 6.3) that the classification performance of the *grafted* and *optimum* classifiers improves as the attack intensity gets higher, while the classification performance of the *optimum* are the best in all scenarios, as expected. This indicates that the both algorithms, "graft" and "re-use", can be used as the bootstrapping approaches to train classifiers for a new environment with insufficient attack information. The results of the second experiment (see Section 6.4) show that the *grafted* classifier has a performance edge over the *re-use* classifier, especially in terms of false positive rates, which is highly desirable for an IDS.

CHAPTER 7

RECONNAISSANCE INTRUSION DETECTION SYSTEM

This chapter describes the architectures, the algorithms and the experimental results of the reconnaissance intrusion detection system (RIDS), which is designed to detect *P&S* attacks and to discover the distributed stealthy attack scenarios.

Section 7.1 briefly reviews the existing literature in reconnaissance attack detection. Section 7.2 introduces the architecture of the reconnaissance intrusion detection system (RIDS). Section 7.3 describes the diagram and the algorithms of RAP. Section 7.4 explains the details of the RAC system. Section 7.5 reports the testing results of RAP and RAC on three different *P&S* attack data sets. Section 7.6 summarizes and concludes this chapter.

7.1 Literature Review in Reconnaissance Intrusion Detection

Reconnaissance attacks (also called probing and scanning attacks or *P&S* attacks) are often used by hackers as the starting phase of a series of hostile activities by scanning a set of addresses or ports, looking for vulnerabilities and possible penetrating points. Therefore, it is important for an IDS to be able to accurately detect reconnaissance attacks as early warnings for more serious attacks.

The state of the practice in *P&S* attack detection is surprisingly simple. Nearly all commercial and open-source intrusion detection systems, e.g. NFR [25] and Bro [9], etc, contain primitive modules to detect portscans and other forms of surveillance activities by looking for X "events of interests" for host Y with Z seconds [71]. These techniques can be easily defeated by attackers simply by increasing their scanning interval or by using multiple hosts to simultaneous scanning a site.

Snort [41] uses three portscan detection methods. The first is a signature detection method by matching packets with the pre-specified scan signatures. The second is connection-oriented method by using the simplistic method mentioned in the previous paragraph. As of version 2.0, SNORT starts to include a new preprocessor "flow-portscan" to detect surveillance attacks by scoring the flows accessing unopened or unpopular services and generating alerts when a certain score threshold have been exceeded.

Several research papers have been published to address the issue of scanning detection using probabilistic methods. Staniford et al. [37] proposed algorithms to detect scans based on the probability of accessing each destination host and port. Leckie et al. [72] designed a probabilistic model to detect likely scan sources. Robertson et al. [73] monitored failed connections attempts and used a simple thresholding method to detect scans. Jung et al. [74] also monitored failed sessions and used a threshold random watch algorithm to detect scanning sessions.

7.2 Reconnaissance Intrusion Detection System

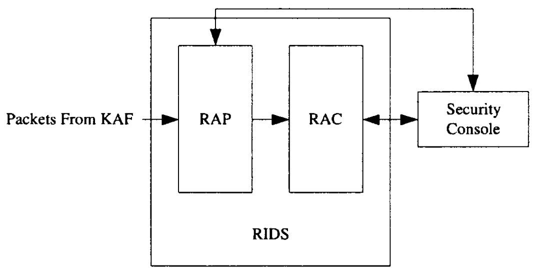


Figure 7.1 Reconnaissance intrusion detection system.

The reconnaissance detection system, Figure 7.1, consists of two functional subsystems:

- Reconnaissance Activity Profiler (RAP): is a session-based intrusion detection module capable of detecting stealthy scanning and probing attacks. The detailed information about RAP will be described in Section 7.3.
- Reconnaissance Activity Correlater (RAC): is an alert-correlation module to correlate the alerts generated by RAPinto attack scenarios and to discover the

distributed stealthy attack scenarios. The detailed information about RAC will be described in Section 7.4

7.3 Reconnaissance Activity Profiler

The RAP module processes the packets from the KAF module, builds sessions, aggregates the sessions based on the client IP addresses, extracts and scores the session features, classifies the features, and reports the alerts to both the security console and to RAC module. As illustrated in Figure 7.2, RAP consists of the following functional sub-modules:

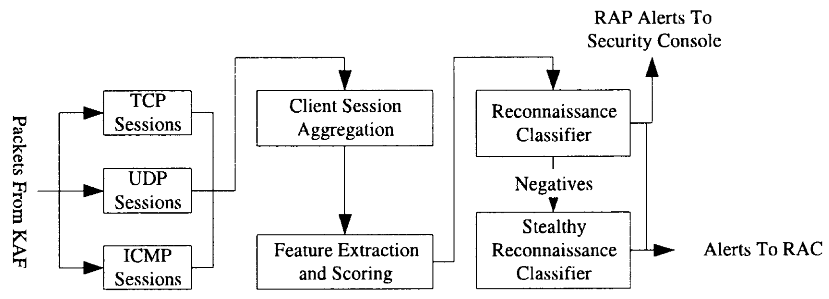

RAP Alerts To Security Console
Packets From KAF
TCP Sessions
UDP Sessions
ICMP Sessions
Client Session Aggregation
Feature Extraction and Scoring
Reconnaissance Classifier
Negatives
Stealthy Reconnaissance Classifier
Alerts To RAC

Figure 7.2 The diagram of the reconnaissance activity profiler.

- **Session Assembly**: assembles the packets into sessions. There are three types of sessions: TCP sessions, UDP sessions, and ICMP sessions. Although UDP and ICMP packets are essentially connectionless, the exchange of packets between clients and servers does convey a lot of valuable information, which can be utilized for the purpose of intrusion detection.

- **Client Session Aggregation**: aggregates all sessions according to their client IP addresses. The aggregated behaviors of the attack clients significantly differ from the behaviors of the normal users. Therefore, client-level session aggregation is very useful for detecting attacks.

- **Session Feature Extraction and Scoring**: extracts the features statistically describing the behaviors of the sessions and their client addresses. It calculates the probability scores these features by comparing the features with the reference models of normal and attack users. The probability scores are measurements indicating how likely for a feature to take the observed values.

- **Reconnaissance Classifier**: classifies the score vectors to detect reconnaissance attacks. The positive (attack) classification outputs are reported to both security console and to following RAC module. The negative (normal) samples are sent to the stealthy reconnaissance classifier for further detection.

- **Stealthy Reconnaissance Classifier**: detects the stealthy reconnaissance activities by further analyzing the negatives of the non-stealthy classifier. It will generate alters to RAC module for whenever stealthy attacks are detected.

7.3.1 Session Monitoring and Categorization

RAP monitors the sessions by rebuilding session dialogs between clients and servers. RAP is designed to monitor three kinds of sessions: TCP, UDP and ICMP. These sessions are aggregated according to their client IP address. For each client, a set of timestamps and counters is used to describe the traffic activities.

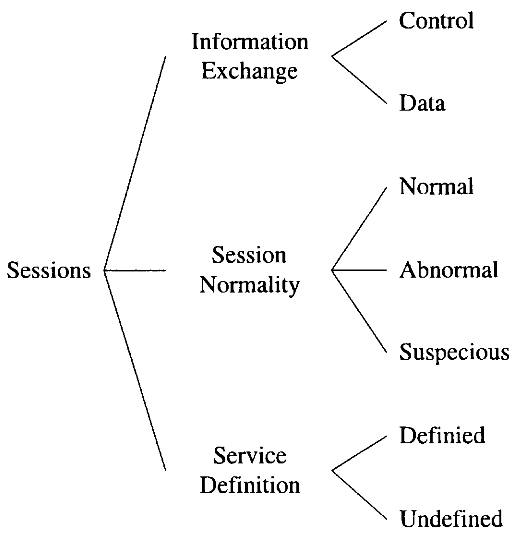


Figure 7.3 The session categories.

The monitored sessions can be categorized into two categories according to their activity patterns:

- **Control Sessions**: are the sessions exchanging control packets only.
- **Data Sessions**: are the sessions exchanging both control and data packets.

Based on control packets and the following state transition paths monitored, sessions are also classified into three different categories:

- **Normal Sessions**: are the sessions without any invalid control packets and anomalous state transitions.
- **Abnormal Sessions**: are the sessions with some anomalous state transitions but no invalid control packets. The possible causes for this kind of sessions are either server busy or packet lost when sniffing.
- **Suspicious Sessions**: are the sessions with invalid control packets. Most commonly these sessions are launched by attacks.

Sessions can also be classified according to their service definition:

- **Defined Sessions**: are sessions accessing the services that are open for normal users.
- **Undefined Sessions**: are sessions accessing either non-existing services or services not open to regular users.

Similar to the anomalous score described by Standiford et al. [37], RAP calculates the session anomalous entropies $E(s)$ using Equation 7.1:

$$E(s) = -log\left[(1-\alpha)\frac{P(s)}{P_{max}} + \alpha\right] \tag{7.1}$$

Where $P(s)$ is the likelihood for a service s to be accessed; P_{max} is the maximum likelihood among all services; α is a predefined small constant coefficient to avoid $log0$ error, which is defined as 0.05 in our system.

7.3.2 RAP Events and Statistical Features

Whenever the state of a session changes, this session together with its correspondent client aggregation will be inspected by RAP for possible anomalous activities. A vector of session features, which statistically describes the session and client activities, are extracted and compared with the reference models.

A RAP **event** is defined as *a session state transition.* Different features are extracted from sessions of different protocols. The detailed description about these features is given in Appendix A.2.

7.3.3 Reference Models and Score Metrics

The extracted session features are then scored based on the information of the reference models. The reference models are in fact probability density functions (PDF) of the feature values. Differing from the reference models of HIDE, the reference models of both normal users and attackers are maintained in RAP. Based on the nature of these features, these reference models can be categorized into two categories:

- Categorical Reference Models: are the reference models of features with discrete values. Two sample categorical reference models are plotted in Figure 7.4. Figure 7.4(a) is the reference model of normal users and Figure 7.5(b) is the reference model of attackers.
- Continuous Reference Models: are the reference models of features with continuous values. Two sample continuous reference models are plotted in Figure 7.5. Figure 7.5(a) is the reference model of normal users and Figure 7.5(b) is the reference model of attackers.

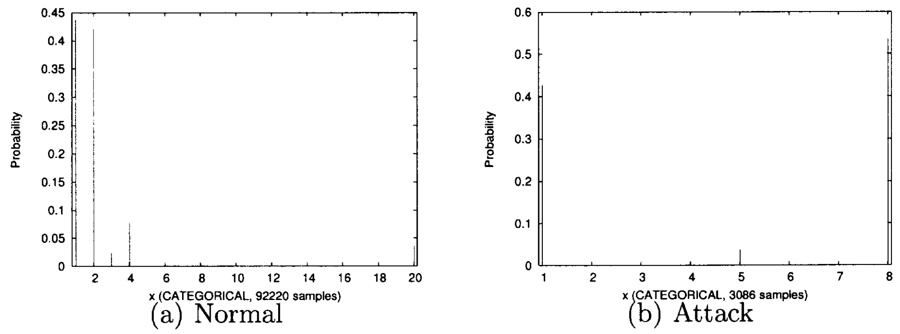
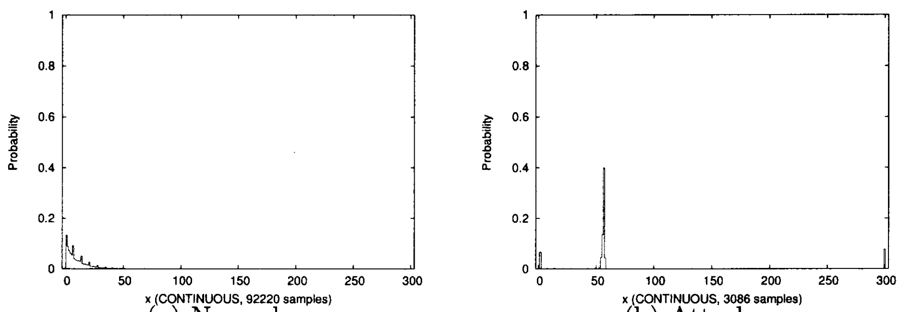


Figure 7.4 Sample categorical reference models.

Two different score metrics are tested. Let x be the feature value, $p_n(x)$ be the probability of x in the normal reference model, $p_a(x)$ be the probability of x in the attack reference model, $p_{n,max}$ be the maximum probability of the normal reference model, and $p_{a,max}$ be the maximum probability of the attack reference model.

The score metric 1 is defined in Equation 7.2.

$$s_1(x) = 2\frac{p_n(x)}{p_{n,max}} - 1 \tag{7.2}$$



(a) Normal (b) Attack

Figure 7.5 Sample continuous reference models.

The score metric 2 is given in Equation 7.3.

$$s_1(x) = \frac{p_n(x)}{p_{n,max}} - \frac{p_a(x)}{p_{a,max}} \tag{7.3}$$

From the above equations, it can be seen that the score metric 1 essentially is an anomaly detection approach by comparing feature values with the known normal patterns, while the score metric 2 is a hybrid signature-anomaly detection approach by utilizing both known normal and known attack knowledge. For simplicity, the score metric 1 will be referred as "METRIC1" and the score metric 2 will be referred as "METRIC2".

7.3.4 Session Classifier

The Back-Propagation Neural Networks are used as the session classifiers of RAP. The inputs of the classifiers are the session feature score vectors and the outputs are the classification results.

7.4 Reconnaissance Activity Correlater

The RAC module builds attack scenarios from the RAP alerts and tries to discover the distributed attack scenarios. The diagram of RAC is shown in Figure 7.6, which consists of the following sub modules:

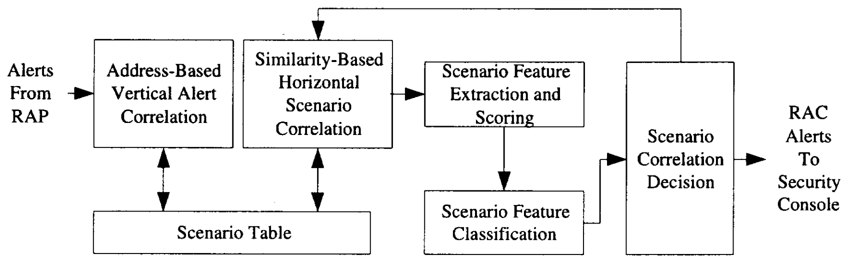

Alerts From RAP
Address-Based Vertical Alert Correlation
Similarity-Based Horizontal Scenario Correlation
Scenario Feature Extraction and Scoring
Scenario Feature Classification
Scenario Table
Scenario Correlation Decision
RAC Alerts To Security Console

Figure 7.6 Reconnaissance activity correlation.

- **Address-Based Vertical Alert Correlation**: correlates RAP alerts vertically, which means all alerts with the same client/server addresses are correlated together.
- **Similarity-Based Horizontal Scenario Correlation**: horizontally correlates attack scenarios, which means that scenarios that are temporally and statistically similar will be fused (correlated) together into one scenario.
- **Scenario Feature Extraction and Scoring**: extracts the statistical features measuring the similarity between two scenarios. These features are then compared with the known reference models.
- **Scenario Feature Classification**: classifies the feature scores and forwards the scores to scenario fusion decision module.
- **Scenario Correlation Decision**: decides whether two scenarios are belong to one distributed attack scenario and hence should be fused together.

7.4.1 Address-Based Vertical Alert Correlation

The alert correlation algorithm of RAC differs from the existing attack fusion algorithms in that it divides the correlation task into two steps: *vertical alert correlation* and *horizontal scenario correlation*. The underlying assumption is that the alerts, which have the same client/server addressed pair and are temporally close, always belong to the same attack scenario. This assumption is straightforward and true for both the DARPA data and the data collected from the CONEX TESTBED network.

The vertical alert correlation is invoked whenever RAC receives a new alert from RAP; the horizontal alert correlation is more complicated and is invoked only at constant time interval. By splitting the attack fusion problem into two sub problems and invoking related modules in different manners, the fusion accuracy can be maximized by removing the unnecessary uncertainties and the mean time the computation cost of alert correlations can be minimized.

When a new alert arrives from the RAP module, the vertical alert correlation module first searches the scenario table for a scenario with the same client/server address pair as the alert. If the scenario is not found, a new scenario is created; otherwise, the statistics of corresponding scenario is updated by the new alert. Scenarios will be purged from the scenario table after a certain inactive period.

7.4.2 Similarity-Based Horizontal Scenario Correlation

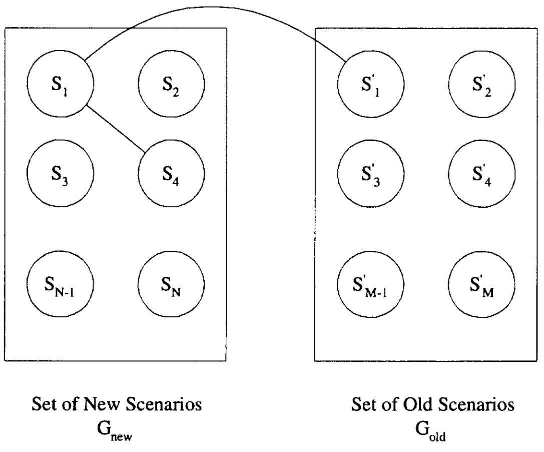


Figure 7.7 Sample scenario sets.

As stated in the previous subsection, the horizontal scenario correlation function is invoked at a constant time interval, e.g. every 60 seconds. When invoked, it first divides all the existing scenarios into two sets: the first set G_{new} includes the scenarios that have been updated since the last horizontal correlation; the second set G_{old} includes the scenarios that were not updated since the last correlation.

The scenarios in Set G_{new} are compared with both the other scenarios in G_{new} and the scenarios in Set G_{old} one by one to determine whether they actually are correlated. For the sample scenarios in Figure 7.7, the binding strengths between scenario S_i in set G_{new} is and the rest scenarios in G_{new} and G_{old} are calculated as a full search for possible correlated scenarios. However, no separate full search will be conducted for the scenarios in Set G_{old}, because no new information is available for these "old" scenarios.

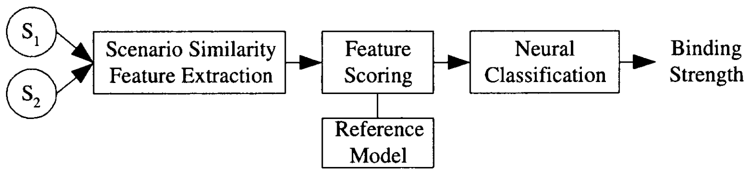


Figure 7.8 The horizontal scenario correlation.

The flow chart to calculate scenario binding strength is shown in Figure 7.8. A vector of statistical features describing the similarity between the two scenarios is extracted. This feature vector is then scored using feature scoring metrics and the learned reference information. A neural network is used to classify the feature scores into a scenario binding strength, which is a comprehensive measurement about the overall similarity of the two scenarios in question. The binding strength is a continuous value, ranging from -1 (same) to 1 (different).

7.4.3 RAC Events and Statistical Features

In RAC, an event is defined as an alert generated by RAP. Totally 28 statistical features are monitored by RAC to describe the statistical similarities between two scenarios. A detailed description of the extracted scenario features is given in Appendix A.3. The feature scoring algorithms are the same as the scoring algorithms of session features described in Section 7.3.3.

7.4.4 Scenario Classification

A Back-Propagation neural network is used as the scenario classifier. The inputs of the classifier are the scenario feature score vectors from the "Feature Scoring" module in Figure 7.8. The outputs of the classifier are the scenario binding strengths.

7.4.5 Correlation Decision

The decision of scenario correlation has profound impact on the system performance. A false decision to mistakenly merge two uncorrelated scenarios could corrupt all the results after that. The purpose of this algorithm is to design an effective decision-making approach, with as few false alarms as possible, on fusing two scenarios.

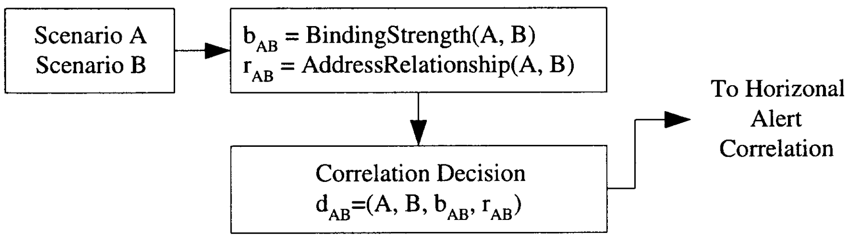

Scenario A
Scenario B
b_{AB} = BindingStrength(A, B)
r_{AB} = AddressRelationship(A, B)
Correlation Decision
d_{AB}=(A, B, b_{AB}, r_{AB})
To Horizonal Alert Correlation

Figure 7.9 Scenario correlation decision.

The block diagram of the scenario correlation decision module is shown in Figure 7.9. In the figure, Scenario A and Scenario B are the two scenarios need to be inspected; $b_{A,B}$ is the binding strength between A and B; $r_{A,B}$ represents the address relationship between A and B, which will be elaborated in Subsection 7.4.5; and $d_{A,B}$ is the correlation decision.

Terminologies

To facilitate the description of the correlation decision algorithm, the definitions of several terminologies are given as below:

- Scenario: is the aggregate of the RAP alerts according to their client/server address pairs. In the rest of this chapter, scenarios are represented by capital letters, e.g. A, B, C ...

- Binding Strength: is the output of the neural classifier to indicate the closeness between two scenarios. It ranges from -1 to 1. The closer is the binding strength to -1, the more similar are the two scenarios. In the rest of this chapter, binding strength between two scenarios A, B at time t is denoted as $b_t(A, B)$.

- Address Relationship: indicates the geographical relationship among the scenarios in the scenario diagram. The definitions of scenario diagram and the scenario address relationship will be given in the next subsection. The symbol $R_{A,B}$ is used to symbolize the address relationship between scenario A and scenario B.

- Correlation Decision: is the output of this algorithm. It equals to either "CORRELATED" or "UNCORRELATED". "CORRELATED" indicates that the two scenarios are correlated and actually belong to one distributed scenario, therefore they should be fused together; "UNCORRELATED" means that the two scenarios look different. Binding decision between scenario A and B at time window t is symbolized as $d_t(A, B)$.

A Sample Scenario Diagram

A sample diagram of different scenarios is illustrated in Figure 7.10. The x-axis is the client IP address; and the y-axis is the server IP address. Seven scenarios (A, B, C, D, E, F and G) are plotted in the diagram as circles.

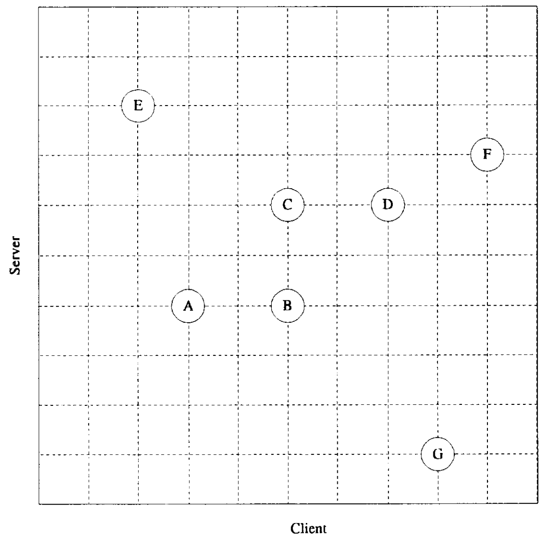


Figure 7.10 A sample diagram of attack scenarios.

The address relationships among the scenarios are defined as below:

- VERTICAL Relationships: are the scenarios that have the same client IP address, e.g. B and C.
- HORIZONTAL Relationships: are the scenarios that have the same server IP address, e.g. A and B, and C and D.
- CROSSING Relationships: are the scenarios that do not share the same client/server IP addresses, e.g. A and C, A and D, B and D, E and F, E and G, etc.

The VERTICAL address relationship occurs when an attacker scan two servers simultaneously. This kind of attacks is very common in both stealthy and none-stealthy attack scenarios and can be easily detected with high accuracy. Therefore, the RAC system should be able to identify this relationship and fuse the vertical-neighboring scenarios promptly.

The HORIZONTAL address relationship happens when attackers launch coordinated attacks scanning server. This kind of attacks is not common but does happen in mostly stealthy attack scenarios. RAC can detect the horizontally distributed scenarios but with less certainty. Therefore, it should be cautious when making the decision to fuse the horizontal scenarios.

The CROSSING address relationship happens mostly among irrelevant attack scenarios or even among false positives. Only in very rare cases, attacker may launch distributed attacks from different sources probing different victim machines. Therefore, it should be very conservative in fusing scenarios with this kind of address relationship.

The RAC Correlation Decision Algorithm

Based on the above observations, a RAC correlation decision algorithm is devised and visualized in Figure 7.11. In the figure, t is the discrete time window index; C_V, C_H and C_C are the three constants indicating how many consecutive

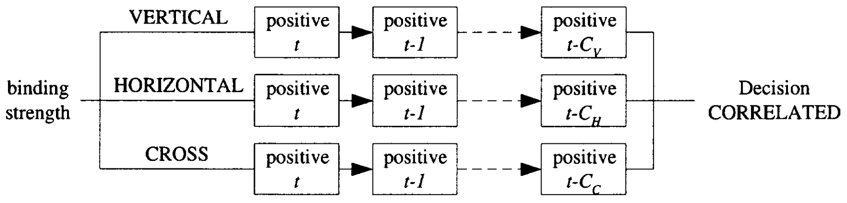


Figure 7.11 The RAC correlation decision algorithm.

"positive/correlated" binding observations, which is evaluated as whether the neural network output is lower than a certain predefined threshold, the RAC system needs before making a "CORRELATED" decision and fusing the two scenarios.

The basic idea of the RAC decision algorithm is to be conservative, when correlating/fusing two scenarios, by using multiple consecutive "positive" observations so that the isolated "false positives" can be filtered and avoided. By adjusting the three constants, C_V, C_H and C_C, the RAC differentiates the likelihood that a "positive" binding strength is actually a "true positive" based on the address relationship between the two scenarios. In this way, a balance between the correlation accuracy and promptness can be achieved and easily configurable by administrators.

7.5 Experimental Results

Three different data sets, which are the DARPA'98 data set, the CONEX TESTBED non-stealthy data set and the CONEX TESTBED stealthy data set, have been used to test the performance of RAP and RAC.

RAC Evaluation Criteria

Same as before, the false positives, false negatives, and misclassifications can be used to evaluate the performance of RAC, but the definitions of these three measurements are slightly different from before.

- False Positives: are the instances that two uncorrelated scenarios are mistakenly classified as correlated.

- False Negatives: are the instances that two correlated scenarios are mistakenly classified as uncorrelated.
- Misclassifications: are the instances of either false positives or false negatives.

7.5.1 Results on the DARAPA'98 Data Set

In this experiment, the RAP is trained to detect the TCP-related reconnaissance attacks ("portsweep", "nmap" and "satan") within the DARPA'98 data set. Five days of traffic (*y98w3d3*, *y98w4d4*, *y98w6d2*, *y98w6d4* and *y98w7d2*) within the DARPA'98 training data set are tested. The detection results are listed in Table 7.1.

Table 7.1 The RAP Detection Results on the DARPA'98 Data

	METRIC1	METRIC2
# of Samples	286346	286346
# of Normal Samples	252045	252045
# of Attack Samples	34301	34301
# of False Positives	161	145
# of False Negatives	52	36
# of Misclassifications	213	181
False Positive Rate	0.0639%	0.0575%
False Negative Rate	0.152%	0.105%
Misclassification Rate	0.0744%	0.0632%

From Table 7.1, it can be seen that both Metric 1 and Metric 2 performed very well in detecting reconnaissance attacks with very low misclassification rates. In comparison, Metric 2 could have a small performance advantage over Metric 1.

Because there is no distributed reconnaissance attack scenario in the DARPA'98 data, only the vertical correlation algorithm can be tested. The attack scenarios in the *y98w6d2* TCPDUMP trace files are listed in Table 7.2 as examples of the vertical scenarios discovered by RAC.

In the table, the "start time" and the "duration" are the starting time and the duration of the attack scenario; "Server" is the IP address of the victim server;

Table 7.2 Attack Scenarios in the Tuesday of Week 6

	Start Time	Duration	Server	Client	Ssns	T_{sp}
1	07/09/1998 12:04:16	06:04:48	172.16.113.50	202.247.224.89	97	226
2	07/09/1998 12:29:52	00:00:01	172.16.114.50	207.103.80.104	300	0
3	07/09/1998 13:57:20	00:03:12	172.16.114.50	195.115.218.108	9814	0
4	07/09/1998 14:42:08	00:36:16	172.16.112.50	206.48.44.18	1024	2.1

"Client" is the IP address of the attacker; "Ssns" is the number of attack sessions detected; and "T_{sp}" is the average period of the attack sessions (in seconds/scan).

From Table 7.2, it can be seen that the TCPDUMP trace file of the Tuesday of Week 6 contains the following four independent scanning scenarios.

1. Scenario 1 is a stealthy "portsweep" scenario with 97 scans at the average scan interval of 226 seconds;
2. Scenario 2 is a clear "portsweep" scenario with 300 scans at the scan interval close to 0 seconds;
3. Scenario 3 is a clear "satan" attack;
4. Scenario 4 is a "portsweep" scenario with 1024 scan sessions at the rate of 2.1 seconds per scan.

7.5.2 Results on the CONEX TESTBED Non-Stealthy Attacks

Non-stealthy *P&S* attacks are simulated within the CONEX TESTBED network (see Figure 3.7). In the simulation, the non-stealthy *P&S* attacks are defined as distributed attack scenarios with short to medium probing intervals ranging from 2 seconds to 2 minutes. The RAP detection results on the non-stealthy data set, using both METRIC1 and METRIC2, are listed in Table 7.3.

The results in Table 7.3 show that RAP can reliably detect non-stealthy *P&S* attacks with very low false negative rate and zero false positive rate.

Table 7.4 gives the detection performance of RAC on the non-stealthy data set. From the table, it can be seen that the METRIC2 has much lower false positive rates than the METRIC1. This is highly desirable, because, as stated in Section 7.4.5, a

Table 7.3 RAP Detection Results on the Non-Stealthy Data Set

	METRIC1	METRIC2
# of Samples	30223	30223
# of Normal Samples	12806	12806
# of Attack Samples	17421	17421
# of False Positives	0	0
# of False Negatives	4	4
# of Misclassifications	4	4
False Positive Rate	0	0
False Negative Rate	0.00230%	0.00230%
Misclassification Rate	0.00132%	0.00132%

Table 7.4 RAC Detection Results on the Non-Stealthy Data Set

	METRIC1	METRIC2
# of Samples	5635	6458
# of Normal Samples	5156	5936
# of Attack Samples	479	522
# of False Positives	92	41
# of False Negatives	2	11
# of Misclassifications	94	52
False Positive Rate	1.78%	0.691%
False Negative Rate	0.418%	2.11%
Misclassification Rate	1.67%	0.805%

false positive (mistakenly merging two uncorrelated scenarios) could corrupt all the results after that.

7.5.3 Results on the CONEX TESTBED Stealthy Attacks

Stealthy *P&S* attacks, which are defined as distributed scanning attacks with medium (starting from 5 minutes) to long (up to 1 hour) scan intervals, are also simulated within the CONEX TESTBED network. The RAP detection results on stealthy attacks are tabulated in Table 7.5. The table shows that RAP can detect all stealthy scanning attacks without a single false positive.

Similar to the RAP detection results, the RAC detection results of METRIC1 and METRIC2 are evaluated and listed in Table 7.6. The misclassification rate of

Table 7.5 RAP Detection Results on the Stealthy Data Set

	METRIC1	METRIC2
# of Samples	31668	31668
# of Normal Samples	30390	30390
# of Attack Samples	1278	1278
# of False Positives	0	0
# of False Negatives	0	0
# of Misclassifications	0	0
False Positive Rate	0	0
False Negative Rate	0	0
Misclassification Rate	0	0

Table 7.6 RAC Detection Results on the Stealthy Data Set

	METRIC1	METRIC2
# of Samples	2142	2142
# of Normal Samples	1101	1101
# of Attack Samples	1041	1041
# of False Positives	92	62
# of False Negatives	205	66
# of Misclassifications	297	128
False Positive Rate	8.36%	5.63%
False Negative Rate	19.92%	6.34%
Misclassification Rate	13.97%	5.97%

METRIC2 is once again lower than that of METRIC1. This hints that the METRIC2 could be more accurate in correlating attack scenario. The misclassification rates of the both metrics are relatively high when comparing the results of non-stealthy attacks (see Table 7.4). This indicates that distributed stealthy attack scenarios, which are widely separated in time and the address/port spaces, are much more difficult to be correctly correlated than the non-stealthy scan scenarios.

Table 7.7 lists the performance of RAC decision module at different combinations of C_V, C_H and C_C. The last three columns of the table give the numbers of merging decisions and the numbers of false merging decisions in the format of "number of merging decisions/number of false decisions". The "Vertical" column corresponds to the performance when merging scenarios of VERTICAL

Table 7.7 The RAC Decision Results of CONEX TESTBED Stealthy Data

C_V	C_H	C_C	Vertical	Horizontal	Cross
1	1	1	22/11	7/1	2/0
2	2	2	14/1	8/0	3/1
3	3	3	13/1	8/0	2/1
3	3	4	12/0	8/0	0/0

address relationship; the "Horizontal" column corresponds to the performance when merging scenarios with HORIZONTAL address relationship; and the "Cross" column corresponds to the performance when merging scenarios with CROSS address relationship.

When C_V, C_H and C_C are all set to one, there are 12 false merges out of the total 32 merges. As C_V, C_H and C_C are set to larger values, the number of merges and the number of false merges become smaller. When C_V, C_H and C_C are set to three, three and four, respectively, the number of false merge becomes zero. This indicates that, by selecting proper C_V, C_H and C_C, the RAC decision module can significantly reduce the number of false merges.

7.6 Comparison and Conclusion

Almost all the existing literatures used different data sets, most of which are not publicly accessible. Many of these publications used network logs collected from real networks, which do not have the necessary attack truth. Therefore, a fair comparison on the reconnaissance detection performance among these systems is difficult. Instead, some of the existing results on *P&S* attacks are listed as below:

1. The best *P&S* detection result in the DARPA'98 IDEVAL project can detect "almost 90% of the probes but with very few false alarms" [45]. The DARPA'98 IDEVAL project used "session" as the unit of events.
2. The best *P&S* detection results in DARPA'99 IDEVAL project can detect about 70% of all probes with 10 false alarms per day (see Figure 3 of [46]). The DARPA'99 IDEVAL project used "attack instance" as the unit of events

3. Leckie et al. [72] tested their systems using packet logs from two sources. The event unit they used was "traffic source", which is very similar to the definition of attack scenarios of RAC. According to the paper, the false positive rate was 70%.

The following conclusions can be got from the results shown in the above section:

- RAP is a very promising session-based ID system to detect reconnaissance attacks, stealthy or non-stealthy. Some early results indicate that RAP can detect attacks with high detection rates and very low false alarm rates.
- RAC can reliably discover distributed non-stealthy scanning scenarios with very low misclassification rates. The misclassification rates for the stealthy scenarios are relatively higher that the non-stealthy scenario, but, when considering the difficulty of correlating stealthy scenarios, these results are still regarded as very good.
- The performance of the score metric 2 is consistently better than the score metric 1.

CHAPTER 8

CONCLUSIONS

This dissertation presented the research efforts to apply PDF statistics and neural network classification in network intrusion detection and alert correlation. The following research topics have been addressed and reported in the dissertation:

8.1 The System Architecture and the Statistical Model of HIDE

HIDE, a hierarchical statistical anomaly intrusion detection systems to detect *DoS* attacks, were presented in Chapter 3. The system had been tested by three different data sets: the OPNET simulation, the DARPA'98 intrusion detection evaluation data and the CONEX TESTBED data. The results indicated that HIDE can reliably detect *DoS* attacks with very false positive and false negative rates.

8.2 HIDE Feature Subset Selection

Experiments to select feature subsets important for HIDE to detect *DoS* attacks had been carried out and report in Chapter 4. The experimental results indicated that, by selecting proper feature subset, HIDE can effectively reduce the number of monitored features from 45 to 8.

8.3 Systematic Research Efforts to Optimize the Performance of HIDE

Various research activities to optimize the performance of HIDE had been performed. The results and conclusions are reported in Chapter 5, which include:

1. **The study of the PDF binning schemes**

   The study on three different PDF binning algorithms, which are used to build and to represent probability density functions in computer systems, had been carried out and reported in Section 5.1. The results showed that the "percentile" partition with 16 bins could be the optimum choice for the trade off between the computation complexity and the system performance.

2. **Compressing PDFs using wavelet compressions**

   Four different wavelet algorithms, *haar*, *sym2*, *coif3* and *db4*, with various compression ranges, were tested to compress PDFs generated by HIDE (see Section 5.2). The results showed that all four wavelet bases could reliably compress the PDFs from 64 bin values to 16 to 28 wavelet coefficients (compression range 3), without major performance deterioration. Among the four wavelet algorithms, the *sym2* wavelet family seemed to perform the best.

3. **The study of similarity metrics**

   The effectiveness of four different similarity metrics, which are used to measure the similarity distances between two PDFs, has been investigated in Section 5.3. By comparing and analyzing the experimental results of these metrics, the AKS (see Equation 3.3) was chosen as the similarity metric of HIDE.

4. **The study of neural network classifiers**

   As the kernels of intrusion detection systems, classifiers have profound impacts on the system performance and efficiency. Experiments had been conducted and reported in Section 5.4 to evaluate five different types of neural networks with various numbers of hidden neurons. The results indicated that BP and PBH networks provide more efficient classification results than the other alternatives.

8.4 Methods of Classifier Training in a Production Network

Two different algorithms to train ID classifiers in a production network using the information gathered from a test network, *graft* and *re-use*, had been studied and reported in Chapter 6. The results showed that the both algorithms can be used as the bootstrapping approaches to train classifiers for a new environment with insufficient attack information.

8.5 Reconnaissance Intrusion Detection System

The architectures and the detection algorithms of the proposed reconnaissance intrusion detection system (RIDS), which consists of two consecutive modules, RAP and RAC, had been presented in Chapter 7 to detect *P&S* attacks and to discover distributed scan scenarios. The results showed the systems can reliably detect reconnaissance attacks and discover distributed reconnaissance scenarios.

APPENDIX A

MONITORED STATISTICAL FEATURES

This appendix gives the detailed descriptions on the statistical features monitored by HIDE, RAP and RAC.

A.1 The HIDE Features

The HIDE is capable of monitoring traffic into and out of the protected network. The HIDE statistical features can be categorized based on the protocols of network traffic.

A.1.1 The IP Features

Six IP traffic parameters can be monitored by HIDE.

1. **IP Packet Length** measures the averages and the distributions of the IP packet lengths within a time window. For simplicity, this parameter is symbolized as "ip-pkt-len" afterward.

2. **IP Packet Rate** measures the averages and the distributions of the packet rates of all observed IP packets within a time window. This feature will be symbolized as "ip-pkt-rate" afterward.

3. **IP Byte Rate** measures the averages and the distributions of the byte rates of all observed IP packets within a time window. This feature will be symbolized as "ip-byt-rate" afterward.

4. **IP Fragment Rate** measures the averages and the distributions of the packet rates of all observed IP fragments within a time window. This feature will be symbolized as "ip-frag-rate" afterward.

5. **IP Defragmentation Error Rate** measures the averages and the distributions of the IP defragmentation error rates occurred within a time window. This feature will be symbolized as "ip-defrag-error" afterward.

6. **IP Checksum Error Rate** measures the averages and the distributions of the IP checksum error rates occurred within a time window. This feature will be symbolized as "ip-csum-error" afterward.

A.1.2 The TCP Features

The HIDE is capable of monitoring 14 different kinds of TCP-related traffic features.

1. **TCP Invalid Packet Rate** measures the averages and the distributions of the rates of the TCP packets with invalid combinations of TCP control flags. This feature will be symbolized as "tcp-pkt-invalid".

2. **TCP Packet Length** measures the averages and the distributions of the lengths of IP packets within a time window. This feature will be symbolized as "tcp-pkt-len".

3. **TCP Packet Rate** measures the averages and the distributions of the TCP packet rates within a time window. This feature will be symbolized as "tcp-pkt-rate".

4. **TCP SYN Packet Rate** measures the averages and the distributions of the rates of TCP control packets with SYN flag set within a time window. This feature will be symbolized as "tcp-syn-pkt-rate" afterward.

5. **TCP FIN Packet Rate** measures the averages and the distributions of the rates of TCP control packets with FIN flag set within a time window. This feature will be symbolized as "tcp-fin-pkt-rate" afterward.

6. **TCP RST Packet Rate** measures the averages and the distributions of the rates of TCP control packets with RST flag set within a time window. This feature will be symbolized as "tcp-rst-pkt-rate" afterward.

7. **TCP Connection Open Rate** measures the averages and the distributions of the TCP connection open rates within a time window. This feature will be symbolized as "tcp-con-new-opened" afterward.

8. **TCP Connection Close Rate** measures the averages and the distributions of the TCP connection close rate within a time window. This feature will be symbolized as "tcp-con-new-closed" afterward.

9. **TCP Connection Abort Rate** measures the averages and the distributions of the TCP connection abort rate (connection closed by RESET or TIMEOUT other than normal three-way hand shaking) within a time window. This feature will be symbolized as "tcp-con-new-aborted" afterward.

10. **TCP Connections from Different Source Address** measures the distributions of TCP connections from different source IP addresses within a time window. This feature will be symbolized as "tcp-con-diff-src" afterward.

11. **TCP Connection to Different Destination Address** measures the distributions of TCP connections to different destination IP addresses within a time window. This feature will be symbolized as "tcp-con-diff-dst" afterward.

12. **TCP Connection Anomalous Entropy** measures the averages and the distributions of the anomalous entropies of all TCP connections within a time window. This feature was first proposed in Staniford [37]. The equation to calculate the connection anomalous entropy is given in Equation 7.1. This feature will be symbolized as "tcp-con-anomalous-entropy" afterward.

13. **TCP Connection Half Opened Ratio** measures the averages and the distributions of the ratio between the half-opened TCP connections and all TCP connections within a time window. This feature will be symbolized as "tcp-con-half-opened-ratio" afterward.

14. **TCP Connection Duration** measures the averages and the distributions of the TCP connection durations within a time window. This feature will be symbolized as "tcp-con-duration" afterward.

A.1.3 The UDP Features

The HIDE can be configured to monitor five different UDP-related traffic features.

1. **UDP Packet Length** measures the averages and the distributions of UDP packets within a time window. This feature is symbolized as "udp-pkt-len" afterward.

2. **UDP Packet Rate** measures the averages and the distributions of UDP packets within a time window. This feature will be referred as "udp-pkt-rate" afterward.

3. **UDP Byte Rate** measures the averages and the distributions of UDP packets within a time window. This feature will be referred as "udp-byt-rate" afterward.

4. **UDP Packets from Different Sources** measures the distributions of UDP packets from different IP addresses. This feature will be referred as "udp-diff-src" afterward.

5. **UDP Packet to Different Destinations** measures the distributions of UDP packets destined to different IP addresses. This feature will be referred as "udp-diff-dst" afterward.

A.1.4 The ICMP Features

The HIDE can monitor seven different ICMP-related traffic features.

1. **ICMP Packet Length** measures the averages and the distributions of ICMP packet lengths within a time window. This feature will be referred as "icmp-pkt-len" afterward.

2. **ICMP Packet Rate** measures the averages and the distributions of ICMP packets within a time window. This feature will be symbolized as "icmp-pkt-rate" afterward.

3. **ICMP Packets from Different Sources** measures the distributions of ICMP packets originated from different IP addresses within a time window. This feature will be symbolized as "icmp-diff-src" afterward.

4. **ICMP Packet to Different Destinations** measures the distributions of ICMP packets destined to different IP addresses within a time window. This feature will be referred as "icmp-diff-dst" afterward.

5. **ICMP Anomalous Echo Replies** measures the averages and the distributions of anomalous ICMP echo replies, which are ICMP echo reply packets without previous echo request packets, within a time window. This feature will be referred as "icmp-anomalous-echo-reply" afterward.

6. **ICMP DUR Packet Rate** measures the averages and the distributions of ICMP DUR (destination-Unreachable) packets within a time window. This feature will be referred as "icmp-dur-pkt-rate" afterward.

A.1.5 Features in the OPNET Data Set

To detect the *UDP flooding* attacks simulated in the OPNET data set (see Section 3.5.1 for detailed information), ten traffic features are monitored by HIDE. They are: in.ip-pkt-len, in.ip-pkt-rate, in.ip-byt-rate, in.ip-diff-src, in.ip-diff-dst, in.udp-pkt-len, in.udp-pkt-rate, in.udp-byt-rate, in.udp-diff-src and in.udp-diff-dst.

A.1.6 Features in the DARPA'98 and CONEX TESTBED Data Sets

Forty-five traffic parameters are monitored by HIDE to detect the *DoS* attacks in the DARPA'98 data set (see Section 3.5.2) and the CONEX TESTBED data set (see Section 3.5.3). They are: in.ip-pkt-len, in.ip-pkt-rate, in.ip-byt-rate, in.ip-frag-rate, in.ip-defrag-error-rate, in.ip-csum-error-rate, in.tcp-pkt-len, in.tcp-pkt-rate, in.tcp-syn-pkt-rate, in.tcp-fin-pkt-rate, in.tcp-rst-pkt-rate, in.tcp-con-new-opened, in.tcp-con-new-closed, in.tcp-con-new-aborted, in.tcp-con-half-opened-ratio, in.tcp-con-duration, in.tcp-con-diff-src, in.tcp-con-diff-dst, in.tcp-con-anomalous-entropy, in.icmp-pkt-len, in.icmp-pkt-rate, in.icmp-byt-rate, in.icmp-diff-src, in.icmp-diff-dst, in.udp-pkt-len, in.udp-pkt-rate, in.udp-byt-rate, in.udp-diff-src, in.udp-diff-dst, out.ip-pkt-len, out.ip-pkt-rate, out.ip-byt-rate, out.tcp-pkt-len, out.tcp-pkt-rate, out.tcp-con-diff-src, out.tcp-con-diff-dst, out.icmp-pkt-len, out.icmp-pkt-rate, out.icmp-diff-src, out.icmp-diff-dst, out.udp-pkt-len, out.udp-pkt-rate, out.udp-diff-src, out.udp-diff-dst and io.icmp-anomalous-echo-reply.

A.2 The RAP Features

The RAP is capable of detecting TCP, UDP and ICMP reconnaissance attacks. Different feature sets are monitored to detect attacks of different protocols.

A.2.1 The TCP Features

In the current implementation, twenty-seven statistical features are monitored by RAP to detect TCP scanning attacks. They are:

1. **Session Action ID**, or tcp-ssn-act-id, is an enumerated number to describe the session transition action, e.g. session opened, session established, and session closed.
2. **Client Number of Sessions**, or tcp-total-ssns, is the total number of sessions from the client.
3. **Client In-bound Session Ratio**, or tcp-in-ssn-ratio, is the ratio between the number of inbound sessions and the total number of sessions.
4. **Client Out-bound Session Ratio**, or tcp-out-ssn-ratio, is the ratio between the number of outbound sessions and the total number of sessions.
5. **Client Open Session Ratio**, or tcp-open-ssn-ratio, is the ratio between the number of half-open sessions and the total number of sessions.
6. **Client Established Session Ratio**, or tcp-estab-ssn-ratio, is the ratio between the number of established sessions and the total number of sessions.
7. **Client Closed Session Ratio**, or tcp-close-ssn-ratio, is the ratio between the number of closed sessions and the total number of sessions.
8. **Client Rejected Session Ratio**, or tcp-reset-ssn-ratio, is the ratio between the number of sessions rejected by servers and the total number of sessions.
9. **Client Normal Session Ratio**, or tcp-normal-ssn-ratio, is the ratio between the number of normal sessions and the total number of sessions.

10. **Client Abnormal Session Ratio**, or tcp-abnorm-ssn-ratio, is the ratio between the number of abnormal sessions and the total number of sessions.

11. **Client Suspicious Session Ratio**, or tcp-susp-ssn-ratio, is the ratio between the number of suspicious sessions and the total number of sessions.

12. **Client Control Session Ratio**, or tcp-control-ssn-ratio, is the ratio between the number of control sessions and the total number of sessions.

13. **Client Data Session Ratio**, or tcp-data-ssn-ssn-ratio, is the ratio between the number of data sessions and the total number of sessions.

14. **Client Defined Session Ratio**, or tcp-defined-ssn-ratio, is the ratio between the number of defined sessions and the total number of session.

15. **Client Undefined Session Ratio**, or tcp-undefined-ssn-ratio, is the ratio between the number of undefined sessions and the total number of sessions.

16. **Client Session Status**, or tcp-mean-ssn-status, is the average session status of all sessions.

17. **Client Average Session Entropy**, or tcp-mean-ssn-entropy, is the average session entropy of all sessions.

18. **Client Average Session Duration**, or tcp-mean-ssn-duration, is the average session duration of all sessions.

19. **Session Direction**, or tcp-ssn-direction, is the direction of the session (0 for inbound sessions; and 1 for outbound sessions).

20. **Session Server Protocol State**, or tcp-server-state, is an enumerated sever state of the session.

21. **Session Client Protocol State**, or tcp-client-state, is an enumerated client state of the session.

22. **Session Status**, or tcp-ssn-status, is an enumerated number to describe the overall session state, which includes both client and server states (0 indicates an "open" session; 1 indicates an "established" session; 2 indicates a "closed" session; and 3 indicates that the session is rejected).

23. **Session Normality**, or tcp-ssn-normality, is an enumerate number for the session normality (0 for normal sessions; 1 for abnormal sessions; and 2 for suspicious sessions).

24. **Session Type**, or tcp-ssn-cntrl-data, is an enumerate number for session types (0 for control session, within which no real data is exchanged; 1 for data sessions).

25. **Session Service Definition**, or tcp-ssn-definition, is an enumerated number indicating whether the service is defined or not (0 for defined services; 1 for undefined services).
26. **Session Entropy**, or tcp-ssn-entropy, is the entropy of the session (see Equation 7.1).
27. **Session Duration**, or tcp-ssn-duration, is the duration of the session.

A.2.2 The ICMP Features

The following features are monitored by RAP to detect ICMP scanning attacks.

1. **ICMP Session Action ID**, or icmp-ssn-act-id, is an enumerated number to describe the ICMP session transition action, e.g. session opened, session reset, and session closed.
2. **Client Number of ICMP Sessions**, or icmp-total-ssns, is the total number of ICMP sessions from the client.
3. **Client In-bound ICMP Session Ratio**, or icmp-in-ssn-ratio, is the ratio between the number of inbound ICMP sessions and the total number of ICMP sessions.
4. **Client Out-bound ICMP Session Ratio**, or icmp-out-ssn-ratio, is the ratio between the number of outbound ICMP sessions and the total number of ICMP sessions.
5. **Client ICMP RESET Session Ratio**, or icmp-reset-ssn-ratio, is the ratio between the number of ICMP sessions, being rejected by ICMP DUR packets from the other side, and the total number of ICMP sessions.
6. **Client ICMP One-Way Session Ratio**, or icmp-one-way-ssn-ratio, is the ratio between the number of one-way ICMP sessions, ICMP sessions with only one-way traffic observed, to the total number of ICMP sessions.
7. **Client ICMP ECHO Session Ratio**, or icmp-echo-ssn-ratio, is the ratio between the number of ICMP ECHO sessions (sessions of ICMP ECHO request / ECHO reply packets) and the total number of ICMP sessions.
8. **Client ICMP RESET ECHO Session Ratio**, or icmp-echo-reset-ratio, is the ratio between the number of ICMP reset ECHO sessions and the total number of ICMP sessions.
9. **Client ICMP One-Way ECHO Session Ratio**, or icmp-echo-one-way-ratio, is the ratio between the number of ICMP one-way ECHO sessions and the total number of ICMP ECHO sessions.

10. **Client ICMP MASK Session Ratio**, or icmp-mask-ssn-ratio, is the ratio between the number of ICMP ADDRESS sessions, sessions of ICMP address mask request/reply packets, and the total number of ICMP sessions.
11. **Client ICMP RESET MASK Session Ratio**, or icmp-mask-reset-ratio, is the ratio between the number of ICMP reset MASK sessions and the total number of ICMP MASK sessions.
12. **Client ICMP One-Way MASK Session Ratio**, or icmp-mask-one-way-ratio, is the ratio between the number of ICMP one-way MASK sessions and the total number of ICMP MASK sessions.
13. **Client ICMP TIMESTAMP Session Ratio**, or icmp-time-ssn-ratio, is the ratio between the number of ICMP TIMESTAMP sessions, sessions of ICMP timestamp request/reply packets, and the total number of ICMP sessions.
14. **Client ICMP RESET TIMESTAMP Session Ratio**, or icmp-time-reset-ratio, is the ratio between the number of ICMP reset TIMESTAMP sessions and the total number of ICMP TIMESTAMP sessions.
15. **Client ICMP One-Way TIMESTAMP Session Ratio**, or icmp-time-one-way-ratio, is the ratio between the number of ICMP one-way TIMESTAMP sessions and the total number of ICMP TIMESTAMP sessions.
16. **Client ICMP Misc. Session Ratio**, or icmp-misc-ssn-ratio, is the ratio between the number of ICMP sessions that do not belong to the above stated sessions and the total number of ICMP sessions.
17. **Client ICMP RESET Misc. Session Ratio**, or icmp-misc-reset-ratio, is the ratio between the number of ICMP reset misc. sessions and the total number of ICMP misc. sessions.
18. **Client ICMP One-Way Misc. Session Ratio**, or icmp-misc-one-way-ratio, is the ratio between the number of ICMP one-way misc. sessions and the total number of ICMP misc. sessions.
19. **ICMP Session Direction**, or icmp-ssn-direction, is an enumerated number indicating the direction of the session (0 for inbound; and 1 for outbound).
20. **ICMP Session Type**, or icmp-ssn-type, is an enumerate number of ICMP session type, e.g. ECHO, MASK, TIMESTAMP, etc.
21. **ICMP Session Status**, or icmp-ssn-status, is an enumerate number of ICMP session status (0 for two-way session; 1 for one-way session; 2 for reset session).
22. **ICMP Session Request/Reply Ratio**, or icmp-ssn-req-rep-ratio, is the ratio between the ICMP request packets and the ICMP reply packets.
23. **ICMP Session Duration**, or icmp-ssn-duration, is the duration of the ICMP session.

A.2.3 The UDP Features

The RAP monitors the following features of UDP clients to detect UDP *P&S* attacks.

1. **Client UDP Session Action ID**, or udp-ssn-act-id, is an enumerate number describing the UDP session transition action.
2. **Client Number of UDP Sessions**, or udp-total-ssns, is the total number of UDP sessions from the client.
3. **Client In-bound UDP Session Ratio**, or udp-in-ssn-ratio, is the ratio between the number of inbound UDP sessions and the total number of UDP sessions.
4. **Client Out-bound UDP Session Ratio**, or udp-out-ssn-ratio, is the ratio between the number of outbound UDP sessions and the total number of UDP sessions.
5. **Client UDP RESET Session Ratio**, or udp-reset-ssn-ratio, is the ratio between the number of reset UDP sessions (sessions rejected by the other side using ICMP DUR packets) and the total number of UDP sessions.
6. **Client UDP One-Way Session Ratio**, or udp-one-way-ssn-ratio, is the ratio between the number of one-way UDP sessions (sessions with only one-way traffic observed) and the total number of UDP sessions.
7. **Client UDP Defined Session Ratio**, or udp-defined-ssn-ratio, is the ratio between the number of inbound UDP sessions accessing "defined" services and the total number of inbound UDP sessions.
8. **Client UDP Average Session Entropy**, or udp-mean-ssn-entropy, is the average entropy of all UDP sessions from the client.
9. **Client Average Session Duration**, or udp-mean-ssn-duration, is the average duration of all UDP sessions.
10. **UDP Session Direction**, or udp-ssn-direction, is the direction of the UDP session (0 for inbound; and 1 for outbound).
11. **UDP Session Status**, or udp-ssn-status, is an enumerated number of UDP session status, e.g. ONE-WAY, TWO-WAY and RESET.
12. **UDP Session Entropy**, or udp-ssn-entropy, is the anomalous entropy of the UDP session.
13. **UDP Session Service Definition**, or udp-ssn-definition, is an enumerated number indicating whether the accessed UDP service is defined or not (0 for defined UDP services; and 1 for undefined services).
14. **UDP Session Duration**, or udp-ssn-duration, is the duration of the UDP session.

A.3 The RAC Features

Twenty-eight scenario features are monitored by RAC to measure the similarity between two scenarios. They are:

1. **Start Time Difference**, or start-time-diff, is the difference between the scenario start times.
2. **Last Alert Time Difference**, or last-alert-time-diff, is the difference of the times of the last alerts.
3. **Duration Ratio**, or duration-ratio, is the ratio between the durations of the two scenarios.
4. **Alert Number Ratio**, or alert-num-ratio, is the ratio between the numbers of alerts of the two scenarios.
5. **Alert Rate Ratio**, or alert-rate-ratio, is the ratio between the alert rates of the two scenarios.
6. **Session Number Ratio**, or ssn-num-ratio, is the ratio between the numbers of alert sessions of the two scenarios.
7. **Session Rate Ratio**, or ssn-rate-ratio, is the ratio between the rates of the alert sessions of the two scenarios.
8. **Defined Service Ratio**, or defined-srv-ratio, is the ratio between the numbers of the accessed "defined" services of the two scenarios.
9. **Undefined Service Ratio**, or undefined-srv-ratio, is the ratio between the numbers of the accessed "undefined" services of the two scenarios.
10. **Defined/Undefined Service Ratio**, or def-undef-srv-ratio, is the ratio between the defined/undefined service rates of the two scenarios.
11. **Scenario Suspicious Ratio**, or susp-ratio, is the ratio between the average scenario suspiciousness, which are the outputs of the RAP classifier.
12. **Scenario Entropy Ratio**, or entropy-ratio, is the ratio between the average scenario entropies.
13. **Server Site Overlapping**, or server-site-overlap, is the ratio between the numbers of equivalent server addresses and the total number of different server addresses.
14. **Server Alert Overlapping**, or server-alert-overlap, is the ratio between the numbers of alerts to the same servers of the two scenarios and the total number of scenarios.

15. **Client Site Overlapping**, or client-site-overlap, is the ratio between the numbers of equivalent client addresses and the total number of different client addresses.
16. **Client Alert Overlapping**, or client-alert-overlap, is the ratio between the numbers alerts from the same clients and the total number of alerts.
17. **Server Network Overlapping**, or server-net-overlap, is the ratio of the maximum number of overlapping "1" bits in the server network masks of two scenarios.
18. **Client Network Overlapping**, or client-net-overlap, is the ratio of the maximum number of overlapping "1" bits in the client network masks of two scenarios.
19. **Sever Port Site Overlapping**, or server-port-site-overlap, is the ratio between the numbers of equivalent server ports and the total number of different server ports.
20. **Server Port Alert Overlapping**, or server-port-alert-overlap, is the ratio between the numbers of alerts to equivalent server ports and the total number of alerts.
21. **Client Port Site Overlapping**, or client-port-site-overlap, is the ratio between the numbers of equivalent client ports and the total number of different client ports.
22. **Client Port Alert Overlapping**, or client-port-alert-overlap, is the ratio between the numbers of alerts from equivalent client ports and the total number of alerts.
23. **Server Address/Port Site Overlapping**, or server-addr-port-site-overlap, is the ratio between the numbers of equivalent server address/port pairs and the total number of different server address/port pairs.
24. **Server Address/Port Alert Overlapping**, or server-addr-port-alert-overlap, is the ratio between the numbers of alerts to equivalent server address/port pairs and the total number of alerts.
25. **Client Address/Port Site Overlapping**, or client-addr-port-site-overlap, is the ratio between the numbers of equivalent client address/port pairs and the total number of different client address/port pairs.
26. **Client Address/Port Alert Overlapping**, or client-addr-port-alert-overlap, is the ratio between the numbers of alerts from equivalent client address/port pairs and the total number of alerts.
27. **Scan Number Ratio**, or scan-num-ratio, is the ratio between the numbers of scans (number of distinctive server address/port pairs) of two scenarios.

28. **Scan Rate Ratio**, or scan-rate-ratio, is the ratio between the scan rates of two scenarios.

APPENDIX B

ATTACK AND BACKGROUND TRAFFIC EMULATION IN THE CONEX TESTBED NETWORK

The CONEX TESTBED network is a network simulation facility setup in the CONEX lab. of NJIT. This network is designed to emulate the network-based attacks within a fully controlled environment, to provide abundant clearly-labeled network traces and system logs to train and test intrusion detection systems, and to evaluate the real-time detection performance of ID systems.

This chapter introduces the topology and the methodologies of background/attack traffic simulation in the CONEX TESTBED network. The topology of the CONEX TESTBED network will be described in Section B.1. Section B.2 outlines the software tools developed to simulate background and attack traffic with the CONEX TESTBED. Section B.3 presents the details of an attack labeling tools, which is designed to generate the detailed attack truth.

B.1 The Network Topology

As shown in Figure B.1, the CONEX TESTBED network includes the following five subnets, interconnected by three layer-3 Ethernet Switches:

1. **Background Subnet**: consists of simulated normal users, who send out service requests to the servers of the victim network. Several programs have been implemented to initiate requests automatically according to the preset traffic configuration so that little human intervention is needed once the programs are properly configured and start running.
2. **Attack Subnet**: consists of attackers, who try to break into the servers in the victim network. Both *DoS* and *P&S* attacks are simulated in the CONEX TESTBED network.
3. **Victim Subnet**: consists of several servers, which are interconnected with a Fast Ethernet. These servers provide services for the normal users and they are

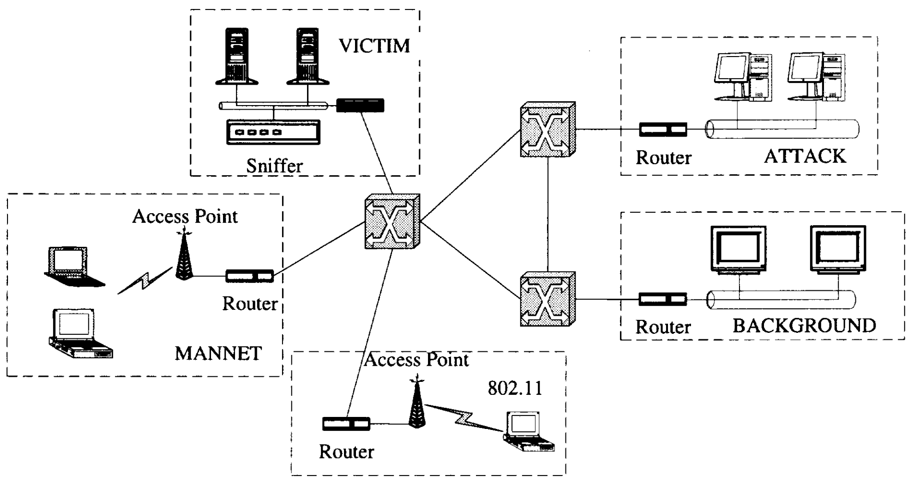


Figure B.1 The topology of the CONEX TESTBED network.

also the targets of the simulated attacks. A packet sniffer is installed within the victim subnet to collect TCPDUMP files for later analysis.

4. **WLAN Subnet**: consists of a wireless access point and several mobile stations, which are interconnected through an IEEE 802.11 wireless LAN. This subnet will be used as both the background and the victim subnets to emulate the increasing 802.11 WLAN traffic of both home and business users.

5. **MANET Subnet**: is an emulated mobile ad-hoc subnet. Instead of setting up a real ad-hoc network, which is technically and financially impractical in a laboratory environment, a "virtual" ad-hoc network, which is emulated with an Ethernet LAN, is used. Intensive R&D efforts are currently undergoing to design and implement the algorithms to emulate the wireless link over a wireless network, to simulate the movements of mobile stations using stationary hosts, and to implement ad-hoc algorithms within a Linux kernel.

At the time when this dissertation is written, the "WLAN" and the "MANET" subnets are still under construction. Therefore, only those "wired" subnets are used in attack simulation and data collection currently.

B.2 The Emulation Tools

This section will concentrate on describing the conceptual and architectural aspects of the background and attack simulation tools.

B.2.1 Background Traffic Emulation

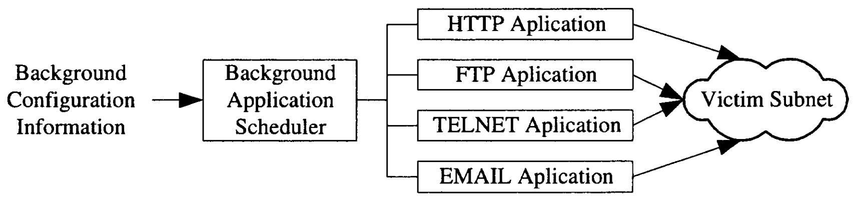


Figure B.2 The background traffic generator.

The background traffic generator, Figure B.2, is the program responsible for launching normal network traffic using commonly used network applications. The generator reads the user-specified background configuration information, such as traffic intensity, the server addresses and the ratio of different applications, etc, and randomly schedules the network applications. To emulate the Internet traffic, an "on/off" traffic model is used to generate traffic with high degree of self-similarities. The background traffic generator can generate traffic using different TCP/IP protocols, e.g. HTTP (web browsing), FTP (file transferring), TELNET (remote accessing) and SMTP (email sending and retrieving). The program has been designed in a way that more applications could be included easily.

B.2.2 Attack Traffic Emulation

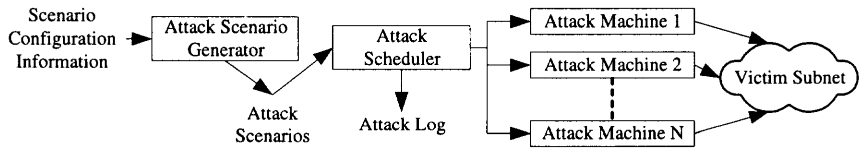


Figure B.3 The attack traffic generator

The flow diagram of the attack traffic emulation procedure is illustrated in Figure B.3. The scenario generator randomly generates the attack scenarios based on the scenario configuration parameters, such as attack duration, attack types, attack hosts, victim targets, etc, which are specified by users. The resulted attack scenarios together with the detailed information about the attack schedules, e.g. attack commands, parameters, targets, scheduled launching time and so on, are stored into a scenario file, which is in turn the input of the attack scheduler.

The attack scheduler is responsible for scheduling and launching attacks based on attack schedule specified in the scenario file. When a scheduled attack is due, the scheduler will commands a subordinate attack machine to launch the attack. Once the attack machine finishes the attack, it will send an attack report, which reports the starting and the ending times as well as the attack status, whether it is successful or failed, back the scheduler. The scheduler will record all this attack information into an attack log, which will be used by the attack labeling tool (see Section B.3) to generate the detailed attack truth.

B.3 The Attack Labeling Tool

Because there is no standard of the formats of attack logs, different attack simulation programs use different logging formats. For example, the DARPA'98 data set used a truth list format to keep the information of attack sessions; the DARPA'99 data set used HTML files to store both the condensed and the detailed attack information; the DARPA'2000 data set used the IDMEF (Intrusion Detection Message Exchange Format) format to specify the high level attack truth; in the CONEX TESTBED network, a XML format with a tag set different from the IDMEF format is used to label both the high-level attack truth and the high-level attack scenario truth.

Due to the diversity and the incompatibilities among these different attack log formats, it is computationally inefficient and difficult to implement for an intrusion detection system to correctly match these logs with various levels of attack details in the network packets. This tool, *Packet Filter*, is designed to provide a generic solution to convert these attack logs of different sources into standard truth formats so that the problem for an IDS to match network packets and the attack truth could be greatly standardized and simplified. The "Packet Filter" needs also be expandable so that the future models to process new attack log formats can be easily plug into this tools without major modifications on the other parts of the source codes.

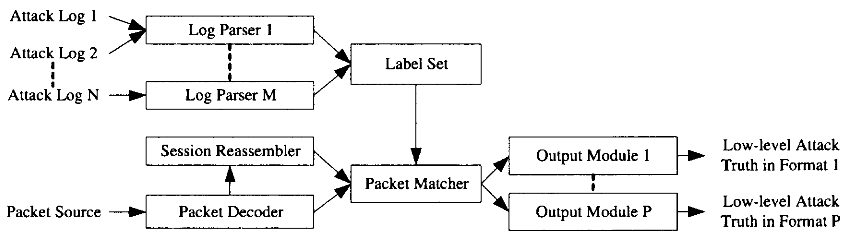


Figure B.4 The system diagram of the packet filter.

The system diagram of the "Packet Filter" is show in Figure B.4, in which includes the following components:

- **The Packet Decoder** decodes the sniffed packets from either a TCPDUMP file or a live network.
- **The Session Reassembler** reassembles the network sessions based on the decoded packets.
- **The Log Parsers** parse the input attack logs from various sources into an internal attack log format and stores the parsed information into the attack label set.
- **The Attack Log** Set is the set of parsed attack logs generated by label parser.
- **The Packet Matcher** matches the packets from the packet decoder and the sessions from the session reassembler against the attack log set.
- **The Output Modules** outputs the matched/mismatched packet/sessions and the attack logs into files or databases in formats specified by users.

Currently, the "Packet Filter" can parse the attack log files in three different formats: the DARPA'98 truth format, the partial attack log format and the CONEX

XML log format. The program can be easily extended to parse attack logs in other formats by developing a new parser and plugging the parser into the system via a unified parser registration interface.

The packet filter can output the low-level attack truth in three different formats: the TCPDUMP binary format, the DARPA'98 truth format and the packet list format. Similar to the log parsers, the output modules are also extendible via a standard class inheritance and registration interface.

REFERENCES

[1] D. Weber, "A taxonomy of computer intrusions," Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 1998.

[2] J. P. Anderson, "Computer security threat monitoring and surveillance," tech. rep., James P. Anderson Co., Fort Washington, PA, April 1980.

[3] D. E. Denning, "An intrusion detection model," *IEEE Transactions on Software Engineering (SE-13)*, pp. 222–232, February 1987.

[4] G. Vigna and R. A. Kemmerer, "Netstat: a network-based intrusion detection approach," in *Proceedings of the 14th Annual Computer Security Applications Conference*, pp. 25–34, 1998.

[5] W. Lee, S. Stolfo, and K. Mok, "A data mining framework for building intrusion detection models," in *Proceedings of 1999 IEEE Symposium of Security and Privacy*, pp. 120–132, 1999.

[6] H. Javitz and A. Valdes, "The NIDES statistical component: description and justification," tech. rep., SRI International, March 1993.

[7] A. Ghosh, J. Wanken, and F. Charron, "Detecting anomalous and unknown intrusions against programs," in *Proceedings of IEEE 14th Annual Computer Security Applications Conference*, pp. 259–267, 1998.

[8] J. Beale and J. Foster, *Snort 2.0 Intrusion Detection.* Syngress Publishing, Inc., 2003.

[9] V. Paxson, "Bro: a system for detection network intruders in real-time," *Computer Networks*, no. 31, pp. 2435–2563, 1999.

[10] T. Lunt and et al., "IDES: an enhanced prototype, a real-time intrusion detection system," tech. rep., SRI International, October 1988.

[11] B. Tjaden, L. Welch, and et al., "INBOUND: the integrated network-based Ohio university network detective service," in *the Proceedings of 4th World Multiconference on Systemics, Cybernetics and Informatics (SCI200)*, (Orlando, Florida), July 2000.

[12] A. Valdes and D. Anderson, "Statistical methods for computer usage anomaly detection using NIDES," tech. rep., SRI International, January 1995.

[13] P. Proctor, "Audit reduction and misuse detection in heterogeneous environments: Framework and applications," in *Proceedings of the 10th Annual Computer Security Applications Conference*, pp. 117–125, December 1994.

[14] R. Heady, G. Luger, A. Maccabe, and Z. Servilla, "The architecture of a network level intrusion detection system," tech. rep., Computer Science Department, University of New Mexico, August 1990.

[15] D. Gollman, *Computer Security.* John Wiley and Son Ltd., 1st ed., 1999.

[16] E. Jonsson, "An integrated framework for security and dependability," in *Proceedings of the New Security Paradigms Workshop*, (Charlottesville, VA, USA), pp. 22–29, September 1998.

[17] L. Halme and K. Bauer, "AINT misbehaving - a taxonomy of anti-intrusion techniques," in *Proceedings of the 18th National Information System Security Conference*, (Baltimore, MD, USA), pp. 163–172, 1995.

[18] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proceedings of the 20th VLDB Conference*, (Santiago, Chile), 1994.

[19] J. Bonifacio and et al., "Neural networks applied in intrusion detection systems," in *Proceedings of The 1998 IEEE Internal Joint Conference on Neural Networks*, pp. 205–210, May 1998.

[20] A. Sung and S. Mukkamala, "Identifying important features for intrusion detection using support vector machines and neural networks," in *Proceedings of the 2003 Symposium on Applications and the Internet (SAINT'03)*, pp. 209–216, 2003.

[21] B. Gao, H. Ma, and Y. Yang, "HMMS (hidden markov models) based on anomaly intrusion detection method," in *Proceedings of the 2002 International Conference on Machine Learning and Cybernetics*, (Beijing, China), pp. 381–385, 2002.

[22] J. Cabrera, B. Bavichandran, and R. Mehra, "Statistical traffic modeling for network intrusion detection," in *Proceedings of 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 466–473, August 2000.

[23] P. Lichodzijewski, A. Nur Zincir-Heywood, and M. Heywood, "Host-based intrusion detection using self-organizing maps," in *Proceedings of the 2002 International Joint Conference on Neural Networks*, (Honolulu, HI), pp. 1714–1719, May 2002.

[24] *Internet Security Systems (ISS).* http://www.iss.net/.

[25] *NFR Security.* http://www.nfr.com/.

[26] W. Frawley, G. Piatetsky-Shapiro, and C. Matheus, "Knowledge discovery in databases: an overview," *AI Magazine*, vol. 13, pp. 57–70, 1992.

[27] N. Ye, X. Li, and S. Emran, "Decision trees for signature recognition and state classification," in *The Proceedings of the IEEE SMC Information Assurance Security Workshop*, (West Point, NY), pp. 189–194, June 2000.

[28] D. Gunneti and G. Ruffo, "Intrusion detection through behavioral data," in *Proceedings of Intelligent Data Analysis (IDA'99)*, (Amsterdam, Netherlands), August 1999.

[29] L. Portnoy, E. Eskin, and S. Stolfo, "Intrusion detection with unlabeled data using clustering," in *Proceedings of ACM Workshop on Data Mining for Security Applications*, pp. 1–14, November 2001.

[30] H. Shah, J. Undercoffer, and A. Joshi, "Fuzzy clustering for intrusion detection," in *The Proceedings of the 12th IEEE International Conference on Fuzzy Systems*, vol. 2, pp. 1274–1278, May 2003.

[31] S. Forrest and et al., *Computer Immune Systems.* http://www.cs.unm.edu/ immsec/.

[32] D. Dasgupta and F. Gonzalez, "An immunity-based technique to characterize intrusions in computer networks," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 281–291, June 2002.

[33] S. Axelsson, "On a difficulty of intrusion detection," in *Proceedings of the 2nd Intl. Workshop on Recent Advances in Intrusion Detection (RAID'99)*, (http://www.raid-symposium.org/raid99/PAPERS/Axelsson.pdf), September 1999.

[34] S. Staniford and et al., "GRIDS - a graphic based intrusion detection system for large networks," in *Proceedings of 19th National Information Systems Security Conference*, pp. 361–370, October 1996.

[35] A. Valdes and K. Skinner, "Probabilistic alert correlation," in *Recent Advances in Intrusion Detection (RAID 2001)*, no. 2212 in Lecture Notes in Computer Science, pp. 54–68, Springer-Verlag, 2001.

[36] O. M. Dain and R. K. Cunningham, "Fusing a heterogeneous alert stream into scenarios," in *Proceedings of the Eighth ACM Conference on Computer and Communications Security*, pp. 1–13, 2001.

[37] S. Staniford, J. Hoagland, and J. McAlerney., "Practical automated detection of stealthy portscans," *Journal of Computer Security*, vol. 10, pp. 105–126, 2002.

[38] *1998 Intrusion Detection Evaluation Data Set.* http://www.ll.mit.edu/IST/ideval/.

[39] Z. Zhang, J. Li, C. Manikopoulos, J. Jorgenson, and J. Ucles, "A hierarchical anomaly network intrusion detection system using neural network classification," in *CD-ROM Proceedings of 2001 WSES International Conference on: Neural Networks and Applications (NNA'01)*, February 2001.

[40] MyDoom. http://vil.nai.com/vil/content/v_100983.htm, January 2004.

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

[42] *KDD cup 1999 data.* http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[43] G. Giacinto and F. Roli, "Intrusion detection in computer networks by multiple classifier systems," in *Proceedings of the 16th International Conference on Pattern Recognition (ICPR2002)*, vol. 2, pp. 390–393, 2002.

[44] Z. Zhang, J. Li, C. Manikopoulos, J. Jorgenson, and J. Ucles, "HIDE: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification," in *Proceedings of the 2nd Annual IEEE Systems, Mans, Cybernetics Information Assurance Workshop*, pp. 85–90, June 2001.

[45] R. P. Lippman and et al., "Results of the darpa 1998 offline intrusion detection evaluation," in *Proceedings of the RAID 1999 Conference*, (West Lafayette, Indiana), September 1999.

[46] R. P. Lippman and et al., "The 1999 darpa off-line intrusion detection evaluation," *Computer Networks*, vol. 34, pp. 579–595, October 2000.

[47] I. Inza, R. Etxeberria, and B. Sierra, "Feature subset selection by Bayesian networks based optimization," tech. rep., University of the Basque Country, Spain, 1999.

[48] R. Martinez, *A Pattern Recognition Feature Optimization using the Visual Empirical Region of Influence Algorithm.* http://www.sandia.gov/imrl/XVisionScience/XVERIFeatureOp.htm, 2002.

[49] A. Miller, *Subset Selection in Regression.* Washington, DC.: Chapman and Hall, 1990.

[50] H. Liu and G. Motoda, *Feature Selection for Knowledge Data Mining.* Kluwer Academic Publishers, 1998.

[51] R. Kohavi and G. John, "Wrappers for feature selection," *Artificial Intelligence*, pp. 273–324, December 1997.

[52] T. Furey and et al., "Support vector machine classification and validation of cancer tissue samples using microarray expression data," *Bioinformatics*, no. 16, pp. 906–914, 2000.

[53] J. Quinlan, *C4.5 Programs for Machine Learning.* Mogan Kaufmann Publisher, Inc., 1993.

[54] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research (JMLR)*, no. 3, pp. 1157–1182, 2003.

[55] A. Dhawan, *Medical Image Analysis.* John Wiley and Sons, Inc., 2003.

[56] C. Borgelt, *http://fuzzy.cs.uni-magdeburg.de/ borgelt/software.html.*

[57] G. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem," in *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 121–129, 1994.

[58] Z. Zhang, C. Manikopoulos, and J. Jorgenson, "Experimental comparisons of binning schemes in representing network intrusion detection data," in *The 36th Conference of Information Sciences and Systems (CISS2002)*, March 2002.

[59] W. Leland and et al., "On the self-similar nature of ethernet traffic," *IEEE/ACM Transactions on Networking*, pp. 1–15, February 1994.

[60] R. Dillon and C. Manikopoulos, "Neural net nonlinear predication for speech data," *IEEE Electronics Letters*, vol. 27, pp. 824–826, May 1991.

[61] S. Haykin, *Neural Network a Comprehensive Foundation.* Macmillan College Publishing Company, 1994.

[62] G. Carpenter and et al., "Fuzzy artmap: an adaptive resonance architecture for incremental learning of analog maps," in *Proceedings of International Joint Conference on Neural Networks*, pp. 759–771, 1992.

[63] NeuralWare Inc., *Neural Computing a Technology Handbook for Neuralworks Professional II/PLUS and NeuralWorks Explorer*, 1998.

[64] Z. Zhang and C. Manikopoulos, "Neural networks in statistical anomaly intrusion detection," *Neural Network Word, International Journal on Non-Standard Computing and Artificial Intelligence*, vol. 11, no. 3, pp. 305–316, 2001.

[65] R. Ogden, *Essential Wavelets for Statistical Applications and Data Analysis.* Birhauser: Boston, 1997.

[66] S. Mallat, *A Wavelet Tour of Signal Processing.* Academic Press, 2 ed., 1999.

[67] I. Daubechies, *Ten Lectures on Wavelets.* Philadelphia, PA: SIAM, 1992.

[68] D. Donoho, M. Duncan, and et al., *WAVELAB 802 for Matlab 5.x.*

[69] Z. Zhang, C. Manikopoulos, J. Jorgenson, and J. Ucles, "Comparison of wavelet compression algorithms in network intrusion detection," in *Proceedings of The International Conference on Computing and Information Technologies (ICCIT2001)*, October 2001.

[70] Z. Zhang, J. Li, C. Manikopoulos, and J. Jorgenson, "Methods of classifier training for anomaly network intrusion detection in a deployed network using test network information," in *Proceedings of the 3rd Annual IEEE Systems, Mans, Cybernetics Information Assurance Workshop (IAW 2002)*, June 2002.

[71] Fyodor, "The art of port scanning," *Phrack 51*, vol. 7, 1997.

[72] C. Leckie and R. Kotagiri, "A probabilistic approach to detecting network scans," in *Proceedings of the Eighth IEEE Network Operations and Management Symposium (NOMS 2002*, (Florence, Italy), pp. 359–372, April 2002.

[73] S. Robertson, E. Siegel, M. Miller, and S. Stolfo, "Surveillance detection in high bandwidth environments," in *Proceedings of the 2003 DARPA DISCEX III Conference*, pp. 130–138, April 2003.

[74] J. Jung, V. Paxson, A. Berger, and H. Balakrishnan, "Fast portscan detection using sequential hypothesis testing," in *Proceedings of IEEE Symposium on Security and Privacy*, (http://www.sds.lcs.mit.edu/papers/portscan-oakland04.html), 2004.