

一個網路異常訊務偵測效能改進之方法

**An Approach for Performance Enhancement of Anomaly Traffic Detection**

研究生：柳竣凱

Student : June-Kai Leou

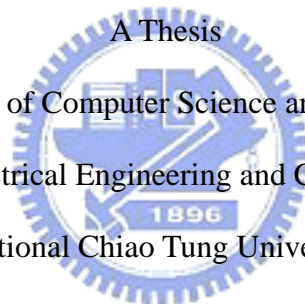
指導教授：陳耀宗 博士

Advisor : Dr. Yaw-Chung Chen

國立交通大學

資訊工程系

碩士論文



Submitted to Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science and Information Engineering

June 2004

Hsinchu, Taiwan, Republic of China

中華民國九十三年六月

# 一個網路異常訊務偵測效能改進之方法

研究生： 柳竣凱

指導教授： 陳耀宗 博士

國立交通大學資訊工程學系

## 中文摘要

現今的攻擊偵測系統大致分成兩個方向，一為以特徵基礎的偵測方式，二為以異常現象為基礎的偵測方式。在本篇論文，我們主要針對異常現象的偵測。我們拿了三個已經設計好的模組來做比較。Packet Header Anomaly Detector 監測封包的 33 個欄位，而 Application Layer Anomaly Detector 著重在應用層的監測，而 Network Traffic Anomaly Detector 使用過濾封包的技術並且位元組為監測的單位。利用封包過濾的技術，大幅降低用以偵測的封包數量。且將不必要再分析的多餘警告給除去，這些方法都可以使得偵測率提升，同時減少警告數量。在實驗中都有詳細的分析。

關鍵字：特徵、異常現象、警告

# **An Approach for Performance Enhancement of Anomaly Traffic Detection**

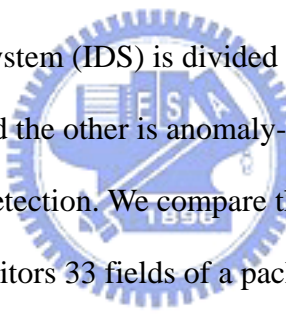
Student: June-Kai Leou

Advisor: Yaw-Chung Chen

Department of Computer Science and Information Engineering

Nation Chiao Tung University

## **ABSTRACT**



The intrusion detection system (IDS) is divided into two aspects generally. One is signature-base detection, and the other is anomaly-base detection. In this thesis, we focus on the anomaly traffic detection. We compare three designed models. Packet Header Anomaly detector monitors 33 fields of a packet, and Application Layer Anomaly Detector emphasizes monitor on the application layer, and Network Traffic Anomaly Detector uses the scheme of traffic filtering and monitors unit by byte-level. By traffic filtering, we can reduce packet need to be detected greatly. And reduce the unnecessary alarms; these approaches can raise the detection rate, and reduce the number of alarms simultaneously. We analyze it in our experiment.

**Keyword:** signature, anomaly, alarm

# Acknowledgement

First, I would express my deep gratitude to my advisor, Prof. Yaw-Chung Chen, for his enthusiastic guidance and continual encouragements. Also, I would deeply appreciate to my seniors, Dr. Yi-Cheng Chan. He gave me beneficial suggestions and encouragement in my research and thesis writing.

Second, I would thank my lab-mate, kent, finphin, kyho, and friends of DSNS lab, oak, joelin, ming, olga, killer, and my roommate, cloudt, Bennett. They tolerate my stupid words and deeds, and say “you are the best” to me when I feel bored. They bring me a nice graduate life.

Finally, I would express my deep indebtedness to my dear family. My mother, and father, and my sisters, and my lovely dog, small white. They give me endless love, and encouragement, and support. I can not do anything without them, so I thank them, and love them forever.



# Contents

<b>Abstract in Chinese</b> .....	i
<b>Abstract in English</b> .....	ii
<b>Acknowledgement</b> .....	iii
<b>Contents</b> .....	iv
<b>List of Figures</b> .....	v
<b>CHAPTER 1</b> .....	1
<b>Introduction</b> .....	1
<b>1.1 A Brief Introduction of Intrusion Detection</b> .....	1
<b>1.2 Thesis Outline</b> .....	3
<b>CHAPTER 2</b> .....	5
<b>Related Works</b> .....	5
<b>2.1 Network intrusion detection</b> .....	5
<b>CHAPTER 3</b> .....	7
<b>Proposed Method</b> .....	7
<b>3.1 Packet Header Anomaly Detector</b> .....	7
<b>3.2. Application Layer Anomaly Detection</b> .....	11
<b>3.3 Network Traffic Anomaly Detector</b> .....	15
<b>3.4 Traffic Filtering</b> .....	16
<b>3.5 Merge Models</b> .....	17
<b>3.6 Reduce Alarms</b> .....	18
<b>CHAPTER 4</b> .....	20
<b>Simulation and Performance Evaluation</b> .....	20
<b>4.1 The 1999 DARPA IDS Evaluation Data Set</b> .....	20
<b>4.2 Compare PHAD with ALAD</b> .....	22
<b>4.3 Traffic Filtering</b> .....	23
<b>4.4 Merge Models</b> .....	24
<b>4.5 Reduce Alarms</b> .....	25
<b>CHAPTER 5</b> .....	30
<b>Conclusion</b> .....	30
<b>5.1 Conclusion</b> .....	30
<b>5.2 Future Work</b> .....	31
<b>REFERENCES</b> .....	32

## List of Figures

Figure 3.1. The PHAD-C32 model after training on week 3. ....	10
Figure 3.2. ALAD models for P (keyword   dest port) for port80, 25, and 21, after training on week 3. ....	13
Figure 4.1. Block diagram of 1999 test bed. ....	20
Figure 4.2. Compare PHAD with ALAD. ....	22
Figure 4.3. PHAD with Traffic Filtering. ....	23
Figure 4.4. Merge PHAD and ALAD. ....	24
Figure 4.5. PHAD with reduce alarms. ....	25
Figure 4.6. ALAD with reduce alarms. ....	26
Figure 4.7. NETAD with reduce alarms. ....	27
Figure 4.8. merge PHAD and ALAD with reduce alarms. ....	28

# CHAPTER 1

## Introduction

### 1.1 A Brief Introduction of Intrusion Detection

Intrusion detection takes an important role in the protection of a network with the growing threat of abuse of network resources. IDS (Intrusion Detection System) are tools used to detect traces of malicious activities that are targeted against the network. A Network IDS (NIDS) monitors packets on the network wire and attempts to discover whether a hacker/cracker is attempting to break into a system. An NIDS may run either on the target machine who watches its own traffic (usually integrated with the stack and services themselves), or on an independent machine promiscuously watching all network traffic (hub, router, probe). NIDS are usually divided into two classes. One is signature based, the other is anomaly based. The System using signature based works as virus scanner and search known, suspicious patterns in their input data. A signature detector, such as SNORT [1] or Bro [2] looks for known attack patterns using rules written by security experts. If some novel attacks are discovered, we must add new rules for these attacks. When traffic consisting of IP datagrams flows across a network, an NIDS is able to capture those packets as they pass by on the wire. A NIDS consists of a special TCP/IP stack that reassembles IP datagrams and TCP streams. It applies some of the following techniques:

(1) **Protocol stack verification** : A number of intrusions, such as

"Ping-O-Death" and "TCP Stealth Scanning" use violations of the underlying IP, TCP, UDP, and ICMP protocols in order to attack the targeted machine. A simple verification system can flag invalid packets, this can include valid, but suspicious, behavior such as severally fragmented IP packets.

**(2) Application protocol verification:** A number of intrusions use invalid protocol behavior, such as "WinNuke", which uses invalid NetBIOS protocol (adding OOB data) or DNS cache poisoning, which has a valid, but unusual signature. In order to effectively detect these intrusions, a NIDS must re-implement a wide variety of application-layer protocols in order to detect suspicious or invalid behavior.

**(3) Creating new loggable events:** A NIDS can be used to extend the auditing capabilities of your network management software. For example, a NIDS can simply log all the application layer protocols used on a host machine. Downstream event log systems (WinNT Event, UNIX syslog, SNMP TRAPS, etc.) can then correlate these extended events with other events on the network.

However, Signature-matching usually has significant limitations. In general, especially when using tight signatures, the matcher has no capability to detect attacks other than those for which it has explicit signature. It is important that signature matcher in general completely miss novel attacks. Unfortunately, new attacks are generated at a brisk pace. And we know that so "tight" signature will miss novel attacks. On the other hand, if we use "loose" signatures, it will raise the major problem of false positives: alert that in fact do not reflect an actual attack, or we say that is a false alarm.

In this thesis, we focus on the anomaly based intrusion detections. Anomaly based detection system watch for deviations of actual from expected behavior and classifies all 'abnormal' activities as malicious. As signature based designs compare their input to known, hostile scenarios they have the advantage of raising virtually no



false alarms. But they have significant drawback of failing to detect variations of known attacks or entirely new attacks. Anomaly detection overcomes the limitation of misuse detection by focusing on normal system behaviors, rather than attack behaviors. This approach always has two phases: in the *training phase*, the behavior of system is observed in the attack-free environment, and machine learning techniques used to create a profile, a “normal” behavior. In the *detection phase*, this profile is compared against the current behavior of the system, and any deviations are marked as potential attacks. Unfortunately, system often exhibit some behavior rightful but unseen before. As a result, it will lead anomaly detection techniques to produce a high degree of false alarms. So anomaly detection has the advantage that no rules are needed to be written, and that it can detect novel attacks. However, it has the disadvantages that it cannot say anything about the nature of the attack, and because normal traffic may also deviate from the model, hence it generates false alarms. On the contrary, anomaly detection can solely bring the suspicious traffic to the attention of a network security expert, who must then figure out what, if any, need to be done. An anomaly detection system such as SPADE [3], ADAM [4], or NIDES [5] models normal traffic, usually the distribution of IP address and ports. Systems that use traffic models monitor the flow of packets. The source and destination IP addresses and ports are used to determine parameters like the number of total connection arrivals in a certain period of time, the inter-arrival time between packets or the number of packet to/from a certain machine. These parameters can be used to detect port scans or denial-of-service attempts.

## 1.2 Thesis Outline

The rest of this thesis is organized as follows. In Chapter 2, we discuss related work in anomaly detection. In Chapter 3, we introduce three models for anomaly detection, PHAD, ALAD, and NETAD. Also we propose methods that can improve the detection performance. In Chapter 4, we use our method proposed in Chapter 3 and analyze the result for each models. Finally, we conclude our work and present some future works in Chapter 5.



## CHAPTER 2

### Related Works

In this Chapter, we introduce the evolution for network anomaly detection.

#### 2.1 Network intrusion detection

Network intrusion detection systems usually use signature detection, matching patterns in network traffic with the patterns of known attacks. This approach is good for detecting intrusion, but has significant disadvantage of being vulnerable to novel attacks. For this reason, we use an alternative approach, anomaly detection, which models a normal behavior of traffic and alerts any deviation from this model as doubtful. Early work in anomaly detection was host based. Forrest et. al. [6] demonstrated that when software errors in UNIX servers or operating system services (*suid root* programs) are exploited in an R2L [13] or U2R attack, that they deviate from the normal pattern of system calls. When compromised, these programs execute code on the behalf of the attacker (usually a shell), rather than the code intended by the developer (such as a DNS name server or print queue manager). Forrest detected these attacks by training an n-gram model ( $n = 3$  to  $6$ ) as the system ran normally. More recent work has focused on better models, such as state machines [4], or neural networks [10]. Solaris makes system call information available through its basic security module (BSM) service for this purpose.

Anomaly detection depends on models of the ‘normal’ behavior of users and applications, and observes the ‘abnormal’ deviations as malicious activity [14]. This approach is different from the misuse detection. In many models, they use the anomaly score to quantify suspicious packets. They quantify the ‘difference’ from the normal behavior of network. Many techniques have been proposed to analyze data stream, such as mining for network traffic [15], statistical analysis for audit records [16]. If a packet has higher score, then it is more suspicious. So if we have good detection rate then we have more efficiency on anomaly detection.



## CHAPTER 3

### Proposed Method

In this Chapter, we introduce three models for anomaly detection: one is packet header anomaly detector (PHAD), one is application layer anomaly detector (ALAD), and the other is network traffic anomaly detector (NETAD). We will take advantage of the skill of NETAD for traffic filtering. Moreover, we use the skill of reducing alarms, and observe the effect of detect result.



#### 3.1 Packet Header Anomaly Detector

Packet Header Anomaly Detector (PHAD) [7] is an algorithm that learns the normal ranges of values for each packet header field at the data link, network, and transport/control layers. But PHAD does not examine application layers protocols like HTTP, DNS, FTP, or SMTP, so it would not detect attacks on servers, although it might detect attempts to hide them from an application layer monitor like *snort* by manipulating the TCP/IP protocols. An important shortcoming of all anomaly detection systems is that they cannot discern intent; they can only detect when an event is unusual, which may or may not indicate an attack. Thus, a system should have a method of ranking alarms by how unusual or unexpected they were, with the assumption that the rarer the event, the more likely it is to be hostile. If this

assumption holds, the user can adjust the threshold to trade off between a high detection rate or a low false alarm rate. PHAD is based on the assumption that events that occur with probability  $p$  should receive a score of  $1/p$ .

PHAD uses the rate of anomalies during training to estimate the probability of an anomaly while in detection mode. If a packet field is observed  $n$  times, and generated  $r$  distinct values, there must have been  $r$  "anomalies" during the training period. If this rate continues, the probability that the next observation will be anomalous is approximated by  $r/n$ . This method is probably an overestimate, since most anomalies probably occur early during training, but it is easy to compute, and it is consistent with the PPMC method of estimating the probability of novel events used by data compression programs (Bell, Witten, and Cleary, 1989).

To consider the dynamic behavior of real-time traffic, PHAD uses a nonstationary model while in detection mode. In this model, if an event last occurred  $t$  seconds ago, then the probability that it will occur in the next one second is approximated by  $1/t$ . Often, when an event occurs for the first time, it is because of some change of state in the network, for example, installing new software or starting a process that produces a particular type of traffic. Thus, we assume that events tend to occur in bursts. During training, the first anomalous event of a burst is added to the model, so only one anomaly is counted. This does not happen in detection mode, so we discount the subsequent events by the factor  $t$ , the time since the previous anomaly in the current field. Thus each packet header field containing an anomalous value is assigned a score inversely proportional to the probability,

$$\text{score}_{\text{field}} = tn/r$$

Finally, we add up the scores to score the packet. Since each score is an inverse

probability, we could assume that the fields are independent and multiply them to get the inverse probability of the packet. But they are not independent, instead, we use a crude extension of the stationary model where we treat the fields as occurring sequentially. If all the  $t_n/r$  are equal, then the probability of observing  $k$  consecutive anomalies in a nonstationary model is  $(r/t_n)(1/2)(2/3)(3/4)\dots((k-1)/k) = (1/k)r/t_n$ . This is consistent with the score  $kt_n/r$  that we would obtain by summation. Thus, we assign a packet score of

$$\text{score}_{\text{packet}} = \sum_{i \in \text{anomalous fields}} t_i n_i / r_i$$

where anomalous fields are the fields with values not found in the training model. In PHAD, the packet header fields range from 1 to 4 bytes, allowing  $2^8$  to  $2^{32}$  possible values, depending on the field. It is not practical to store a set of  $2^{32}$  values for two reasons. First, the memory cost is prohibitive, and second, we want to allow generalization to reasonable values not observed in the limited training data. The approach we have used is to store a list of ranges or clusters, up to some limit  $C = 32$ . If  $C$  is exceeded during training, then we find the two closest ranges and merge them. For instance, if we have the set  $\{1, 3-5, 8\}$ , then merging the two closest clusters yields  $\{1-5, 8\}$ . In the PHAD model, the clusters are set as 32. Because this method (PHAD-C32) gives the best results on the test data that we used.

Field name	r/n	Values
Ether Size	508/12814738	42 60-1181 1182...
Ether Dest Hi	9/12814738	x0000C0 x00105A x00107B...
Ether Dest Lo	12/12814738	x000009 x09B949 x13E981..
Ether Src Hi	6/12814738	x0000C0 x00105A x00107B...

Ether Src Lo	9/12814738	x09B949 x13E981 x17795A...
Ether Protocol	4/12814738	x0136 x0800 x0806 x9000
IP Header Len	1/12715589	x45
IP TOS	4/12715589	x00 x08 x10 xC0
IP Length	527/12715589	38-1500
IP Frag ID	4117/12715589	0-65461 65462 65463...
IP Frag Ptr	2/12715589	x0000 x4000
IP TTL	0/12715589	
IP Protocol	3/12715589	1 6 17
IP Checksum	1/12715589	xFFFF
IP Src	293/12715589	12.2.169.104-12.20.180.101...
IP Dest	287/12715589	0.67.97.110 12.2.169.104-12.20.180.101...
TCP Src Port	3546/10617293	20-135 139 515...
TCP Dest Port	3545/10617293	20-135 139 515...
TCP Seq	5455/10617293	0-395954185 395969583-396150583...
TCP Ack	4235/10617293	0-395954185 395969584-396150584...
TCP Header Len	2/10617293	x50 x60
TCP Flg UAPRSF	9/10617293	x02 x04 x10...
TCP Window Size	1016/10617293	0-5374 5406-10028 10069-10101...
TCP Checksum	1/10617293	xFFFF
TCP URG Ptr	2/10617293	0 1
TCP Option	2/611126	x02040218 x020405B4
UCP Src Port	6052/2091127	53 123 137-138...



UDP Dest Port	6050/2091127	53 123 137-138...
UDP Len	128/2091127	25 27 29...
UDP Checksum	2/2091127	x0000 xFFFF
ICMP Type	3/7169	0 3 8
ICMP Code	3/7169	0 1 3
ICMP Checksum	1/7169	xFFFF

Figure 3.1. The PHAD-C32 model after training on week 3.

As above PHAD examines 33 packet header fields, mostly as defined in the protocol specifications. If it encounters the fields smaller than 8 bits, like the TCP flags, then they are grouped into a single byte field. However, if the fields larger than 4 bytes, like the 6 byte Ethernet addresses, then they are split in half. The philosophy behind PHAD is to build as little protocol-specific knowledge as possible into the algorithm, but we felt it was necessary to compute the checksum fields (IP, TCP, UDP, ICMP), because it would be unreasonable for a machine learning algorithm to figure out how to do this on its own. Thus, we replace the checksum fields with their computed values (normally FFFF hex) prior to processing.

### 3.2. Application Layer Anomaly Detection

The second component of our anomaly detection model is the application layer anomaly detector (ALAD) [9]. Instead of assigning anomaly scores to each packet, it

assigns a score to an incoming TCP connection in a server. TCP connections are reassembled from packets. ALAD, unlike PHAD, is configured knowing the range of IP addresses it is supposed to protect, and it distinguishes server ports (0-1023) from client ports (1024-65535). It does this because it thinks that most attacks are initiated by the attacker (rather than by waiting for a victim), and are therefore against servers rather than clients. We tested a large number of attributes and their combinations that we believed might make good models, and settled on five that gave the best performance individually (high detection rate at a fixed false alarm rate) on the DARPA IDS evaluation data set [8]. These are:

1.  $P(\text{src IP} \mid \text{dest IP})$ , where src IP is the external source address of the client making the request, and dest IP is the local host address. This differs from PHAD in that the probability is conditional (a separate model for each local dest IP), only for TCP, and only for server connections (destination port  $< 1024$ ). In training, this model learns the normal set of clients or users for each host. In effect, it models the set of clients allowed on a restricted service.
2.  $P(\text{src IP} \mid \text{dest IP}, \text{dest port})$ . This model is like (1) except that there is a separate model for each server on each host. It learns the normal set of clients for each server, which may differ out across the servers on a single host.
3.  $P(\text{dest IP}, \text{dest port})$ . This model learns the set of local servers which normally receive requests. It should catch probes that attempts to access nonexistent hosts or services.
4.  $P(\text{TCP flags} \mid \text{dest port})$ . This model learns the set of normal TCP flag sequences for the first, next to last, and last packet of a connection. A normal sequence is SYN (request to open), FIN-ACK (request to close and acknowledge the previous packet), and ACK (acknowledge the FIN). The model generalizes across hosts, but is separate for each port number, because the port number usually indicates the type

of service (mail, web, FTP, telnet, etc.). An anomaly can result if a connection fails or is opened or closed abnormally, possibly indicating an abuse of a service.

5. P(keyword | dest port). This model examines the text in the incoming request from the reassembled TCP stream to learn the allowable set of keywords for each application layer protocol. A keyword means the first word on a line input, i.e. the text between a linefeed and the following space. ALAD examines only the first 1000 bytes, which is sufficient for most requests. It also examines only the header part (ending with a blank line) of SMTP (mail) and HTTP (web) requests, because the header is more rigidly structured and easier to model than the body (text of email messages form uploads). An anomaly indicates the use of a rarely used feature of the protocol, which is common in many R2L attacks.

As described above, the first two rule forms model the set of users of a private (password protected) service, either on a per host or per server basis. The third is intended to detect probes, attempts to access nonexistent hosts or services. The fourth detects malformed or interrupted connections. The fifth models the application layer , and detects some malformed server request.

As with PHAD, the anomaly score is  $tn/r$ , where  $r$  different values were observed out of  $n$  training samples, and it has been  $t$  seconds since the last anomaly was observed. An anomaly occurs only if the value has never been observed in training. For example, Table 2 shows the keyword model for ports 80, 25, and 21, which are the three ports with the highest  $n/r$  values.

For example: ALAD models for P(keyword | dest port) for ports 80, 25, and 21 after training on inside tcpdump files week 3 of the DARPA IDS evaluation data set [8].

Attribute	r/n	Allowed Values
-----------	-----	----------------

80(HTTP)	13/83650	Accept-Charset:
		Accept-Encoding:
		Accept-Language:
		Accept:
		Cache-Control:
		Connection:
		GET
		Host:
		If-Modified-Since:
		Negotiate:
		Pragma:
		Referer:
		User-Agent:
25(SMTP)	34/142610	(34 values...)
21(FTP)	11/16822	(11 values...)

Figure 3.2. ALAD models for  $P(\text{keyword} \mid \text{dest port})$  for port80, 25, and 21, after training on week 3.

The first line of the table says that there are 83,650 TCP connections to port 80, and only 13 different keywords were observed. These keywords are listed in the third column. The total score assigned to a TCP connection is the sum of the  $tn/r$  scores assigned by each of the five components. The keyword model might contribute more than one score because there could be more than one novel keyword.

### 3.3 Network Traffic Anomaly Detector

NETAD (Network Traffic Anomaly Detector), like PHAD, detects anomalies in network packets. However, it differs as follows:

1. The traffic is filtered, so only the start of incoming server requests are examined.
2. Starting with the IP header, we treat each of the first 48 bytes as an attribute for our models--we do not parse the packet into fields.
3. There are 9 separate models corresponding to the most common protocols (IP, TCP, HTTP, etc.).
4. The anomaly score  $tn/r$  is modified to (among other things) score rare, but not necessarily novel, events.

#### NETAD Anomaly Score

We know that PHAD, and ALAD use the anomaly score  $S tn/r$  (summed over the attributes) where  $t$  is the time since the attribute was last anomalous (in training or testing),  $n$  is the number of training instances, and  $r$  is the number of allowed values (up to 256 for NETAD).

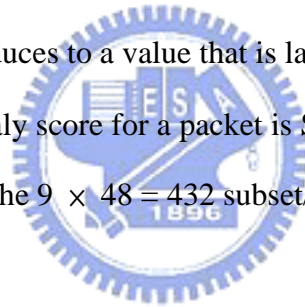
The NETAD model make three improvements to the  $tn/r$  anomaly score. The first improvement, we reset  $n$  (the number of training examples) back to 0 when an anomaly occurs during training. Because the training data contains no attacks, we know that any such anomaly must be a false alarm. The effect is to reduce the weight of this attribute. We call this new score  $tna/r$ , where  $na$  is the number of training packets from the last anomaly to the end of the training period. Note that this is different from  $t$ , which continues to change during the test period. (Like ALAD and LERAD[12], NETAD uses the packet count rather than the real time to compute  $t$ ).

The second improvement is to decrease the weight of rules when  $r$  (the number

of allowed values) is near the maximum of 256. A large  $r$  suggests a nearly uniform distribution, so anomalies are of little value. Thus, we use the anomaly score  $t_{na}$   $(1-r/256)/r$ . For small  $r$ , this is approximately  $t_{na}/r$  as before.

The third improvement, a criticism of PHAD, ALAD, and LERAD is that they ignore the frequency of events. If a value occurs even once in training, its anomaly score is 0. To correct this, we add a second model,  $t_i/(f_i + r/256)$ , where  $t_i$  is the time (packet count in the modeled subset) since the value  $i$  (0-255) was last observed (in either training or testing), and  $f_i$  is the frequency in training, the number of times  $i$  was observed among training packets. Thus, the score is highest for values not seen for a long time (large  $t_i$ ), and that occur rarely (small  $f_i$ ). The term  $r/256$  prevents division by 0 for novel values. It is preferred over a simple count offset (e.g.  $t_i/(f_i + 1)$ ) because for novel events it reduces to a value that is large for small  $r$ .

Thus, the NETAD anomaly score for a packet is  $\sum t_{na} (1 - r/256) / r + t_i / (f_i + r/256)$  where the summation is over the  $9 \times 48 = 432$  subset/attribute combinations.



### 3.4 Traffic Filtering

In NETAD [11], it uses a method that is traffic filtering. This scheme filters out traffic that is uninteresting to us. Because most attacks are initiated against a target victim, so it is usually sufficient to examine only the first few packets of incoming server request. Using this method, we can not only filter out traffic likely to generate false alarm, but also speed up processing. Our approaches are as follows:

- Remove non-IP

- Remove outgoing IP packets not to 172.16.x, 192.168.x, 163.118.135.1
- Remove UDP to high ports
- Remove TCP data packets except near start

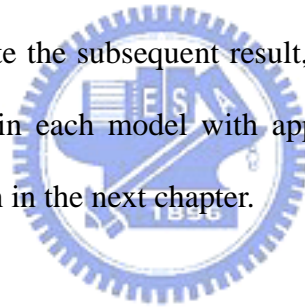
We can use this scheme in other models, for example, packet header anomaly detection. Because in the original PHAD model , the input traffic is 2.70 GB of week 3 inside tcpdump files for training, and 4.64 GB of weeks 4 and 5 inside tcpdump files for testing. After filtering, the training data reduces to 36MB, and the testing data reduces to 69MB. The next step, we evaluate these data instead of the original data. The detail of the experimental statistics will be shown in the next chapter. After our experiment, we find out that the detection rate increases and the processing time decreases significantly. If the original data set is much larger, the efficiency of the scheme will be more notable.



### **3.5 Merge Models**

Because each models has its peculiarity for specific attack detection, so we can combine them for detecting the DARPA inside traffic data. The models mentioned above are useful for detecting anomaly traffic, and give those suspicious packets an anomaly score. Finally, the security experts will analyze the packet with highest score and distinguish whether it is an attack or not. We put the input data into the three models. Because our experiment is based on off-line detection, so we are not very care about the training time and the testing time. Although we do not care about the process time, they make a acceptable performance in our testing computer.

We know that each model generates its result for processing those packets of inside traffic. Since they use different monitoring and scoring methods, therefore, the score of those suspicious packets are different. For example, in our experiment with the 1999 DARPA intrusion data set, the output of the score range is different from each other. The ALAD output has scored 1.009149 decreasing gradually, and the PHAD output has scored 0.748198 decreasing gradually. In actual anomaly detection, it always sorted by score and analyzed by the security experts from the top to the bottom. It is reasonable that the highest score is the most possible malicious packet. So, if we do not adjust the score distribution of these models, for example, in the first 100 false alarms, we should examine the result of ALAD in the great part. This is meaningless for combining two or more models to evaluate. In our experiment, we adjust these score and evaluate the subsequent result, and observe the detection rate. We adjust the highest score in each model with approximation. The details of the experiment will also be shown in the next chapter.



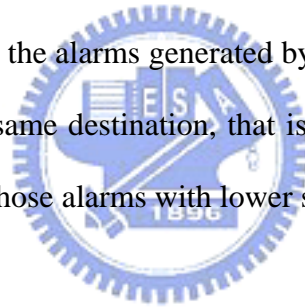
### **3.6 Reduce Alarms**

When we merge these models and evaluate the 1999 DARPA off-line IDS evaluation data set. We will come up against a problem, that is, the alarms will increase greatly. Because each model aiming at each packet will generate a score for a suspicious packet, for the same packet, one model maybe thought that it is suspicious in a high degree, and the others may be not. Consequently, there are different scores represented for one packet. If the data we will evaluate is huge, than the alarms generated will increase dramatically. Therefore the alarms to be analyzed will also



increase greatly. But we know that many alarms are redundant, and we can leave them out of consideration. On the other hand, if we reduce these redundant alarms, it will decrease the false alarms at the same time. Hence, the detection rate in a fixed false alarm will increase relatively.

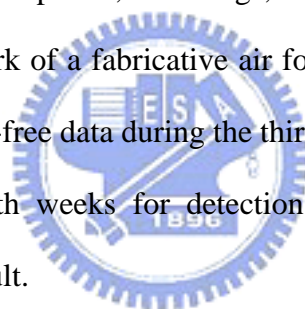
Each model generates different results for each packet if there is one packet which has more than one score. We just take the highest score to represent the packet, and analyze it whether it is an attack or not. This is reasonable, because we need not to analyze the lower score in the same packet. Regarding the same packet, if the highest score we analyze is not an attack, then it wouldn't be an attack for lower score. On the other hand, when we analyze the alarm with highest score and determine that it is an attack, it is reasonable that we need not to analyze those alarms with lower scores. Therefore, we look for the alarms generated by those models, in case it has the same date and time, and the same destination, that is, the same packet, we leave the highest alarm and dispose of those alarms with lower score if they is any.



## CHAPTER 4

### Simulation and Performance Evaluation

In this Chapter, we introduce the 1999 DARPA off-line intrusion data set [8], our experiment uses these data for training and detecting. It consists of network traffic (tcpdump files) collected at two points, BSM logs, audit logs, and file system dumps from a simulated local network of a fabricative air force base over a 5 weeks. In our experiment, we use the attack-free data during the third week for training, and the data during the forth and the fifth weeks for detection. Finally, we use our evaluate program to evaluation the result.



#### 4.1 The 1999 DARPA IDS Evaluation Data Set

DAPRA classifies attacks as probes, denial of service (DOS), remote to local (R2L), user to root (U2R), and other violations of a security policy (Data). The 1999 evaluation was a blind off-line evaluation, as that in 1998, but modified based on suggestions from 1998 and also with major extensions to enhance the analysis and cover more attack types. The Figure 4.1 shows a block diagram of the 1999 test bed.

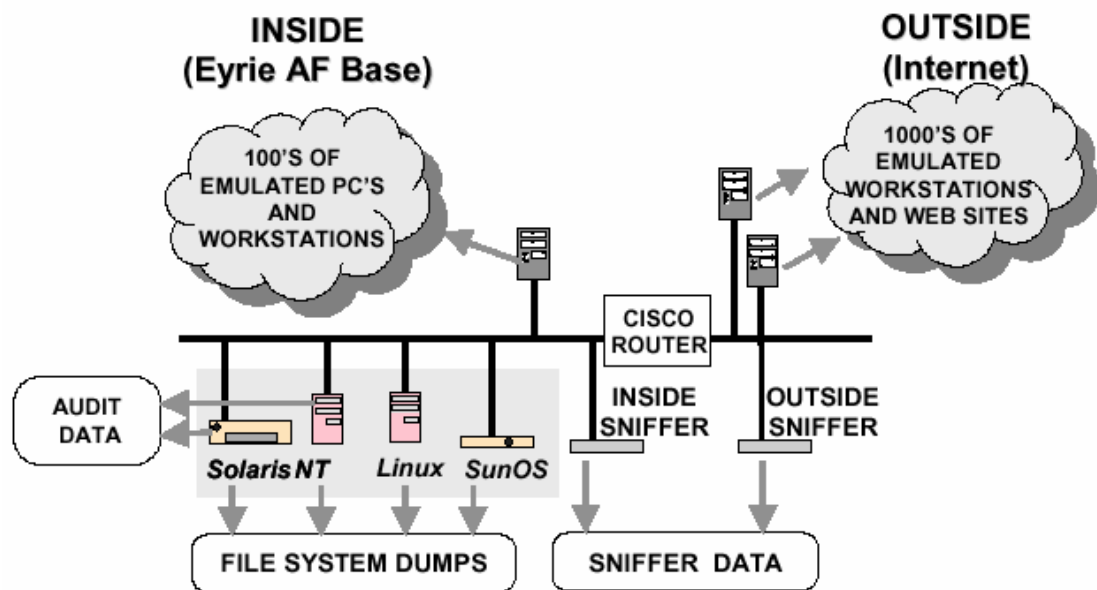


Figure 4.1 Block diagram of 1999 test bed.

Major changes for 1999 are the addition of a Windows NT workstation as a victim, the addition of an inside tcpdump sniffer machine, and the collection of both Windows NT audit events and inside tcpdump sniffing data for inclusion in archival data provided to participants. Not shown in this figure are new Windows NT work-stations added to support NT attacks, new inside attacks, and new stealthy attacks designed to avoid detection by network-based systems tested in 1998. The Windows NT victim machine and associated attacks and audit data were added due to in-creased reliance on Windows NT systems by the military. Inside attacks and inside sniffer data to detect these attacks were added due to the dangers posed by inside attacks.

Stealthy attacks were added due to an emphasis on sophisticated attackers who can carefully craft attacks to look like normal traffic. In addition, two new types of analyses were performed. First, an analysis of misses and high-scoring false alarms was performed for each system to determine why systems miss specific attacks and

what causes false alarms. Second, participants were optionally permitted to submit attack forensic information that could help a security analyst identify important characteristics of the attack and respond. This identification information included the attack category, the name for old attacks, ports/protocols used, and IP addresses used by the attacker.

It contains five weeks tcpdump files and so on, we trains on inside tcpdump files week 3 of the 1999 DARPA IDS evaluation data set, which does not contain any attack. After training, we test on weeks 4 and weeks 5, which contain 185 detectable attacks. In weeks 4 and weeks 5, there are 201 labeled attacks .The inside traffic in week 4, day 2 is missing which contains 12 attacks. There is also one unlabeled attack (apache2) which we found by examining the test data, and there are five external attacks (one queso and four snmpget) against the router which are not visible from inside the local network. This leaves 185 detectable attacks.



## **4.2 Compare PHAD with ALAD**

At first, we compare the PHAD with ALAD model in various false alarms threshold, and we observe the detection result of them.

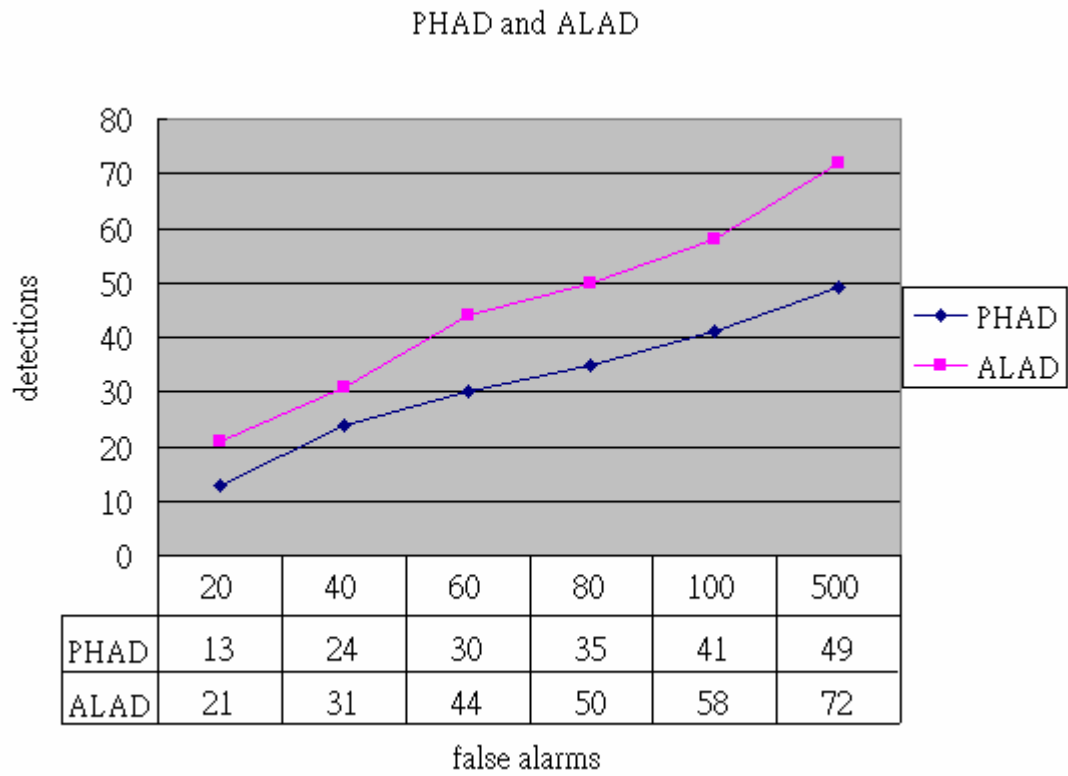
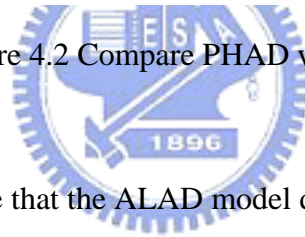


Figure 4.2 Compare PHAD with ALAD.



In Figure 4.2, we observe that the ALAD model detects more than PHAD in each threshold, we also examine the maximum detection numbers in each model. The ALAD model detects 72 when false alarms are 414. In addition, the PHAD model detects 84 when false alarms are 5609. In the maximum detection numbers, PHAD is more than ALAD. But the false alarms reach to 5609, it's too large.

### 4.3 Traffic Filtering

In Section 3.4, we mention the skill of filtering the input data. It is used for the

NETAD model. Now, we use the filtered data as input, and process it by PHAD. We compare it with the original PHAD, as shown in Figure 4.3

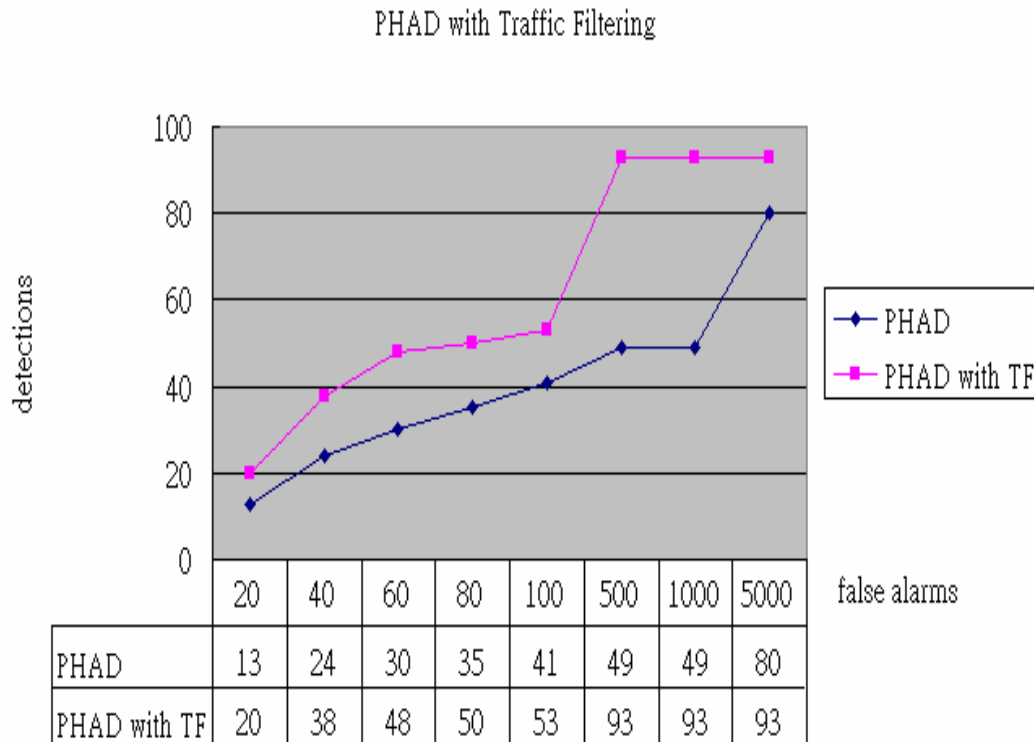


Figure 4.3 PHAD with Traffic Filtering.

In Figure 4.3, we observe that PHAD with Traffic Filtering always detect more anomalies than the original PHAD in each false alarm threshold. The maximum detection number of PHAD with traffic filter is 93 when the false alarms reach to 488. The detection rate is much higher than the original PHAD.

## 4.4 Merge Models

In Chapter 3, we know that each model has its peculiarity for specific attack

detection. According to this reason, we combine two models and observe the number of detection it analyzed. In this case, we use the filtered traffic for PHAD training. Because of the score range they generated are different, we fix the anomaly score of PHAD by adding a value 0.47.

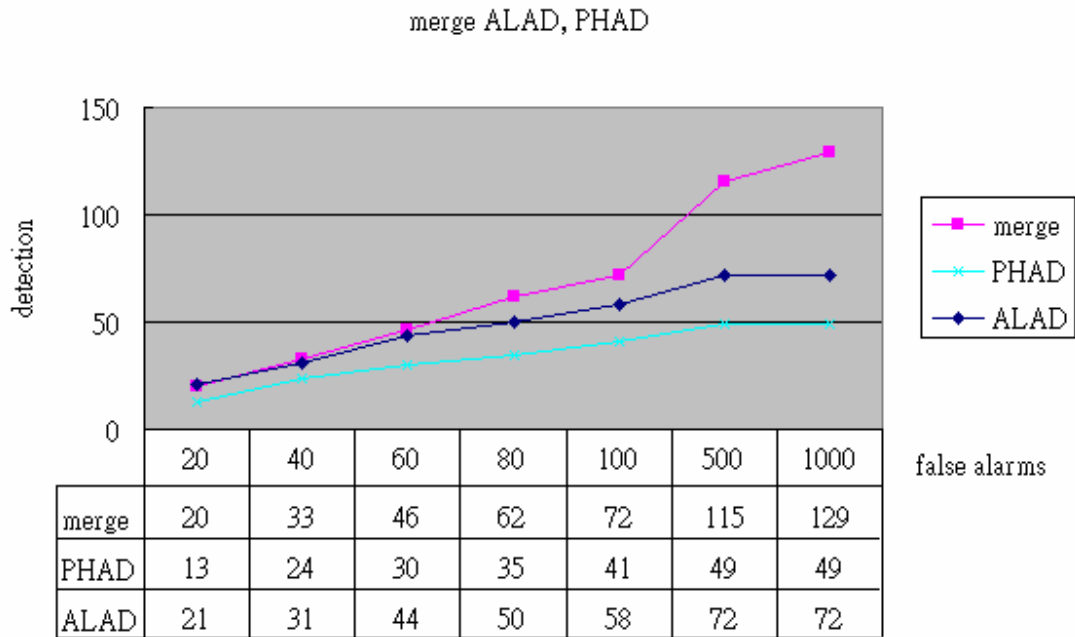


Figure 4.4 Merge PHAD and ALAD.

In Figure 4.4, we can know that we get good detection rate through combining them. The maximum detection number of it is 129 when the false alarms reach to 910. It deserves a claim that the total alarms it generated is 1703. Oppositely, PHAD generate 6260 alarms and get lower detection rate.

## 4.5 Reduce Alarms

In Section 3.6, we know that get rid of unnecessary alarms not only speeds up our analysis process, but also improves the detection rate. We use the scheme to reduce alarms in each model and observe the results.

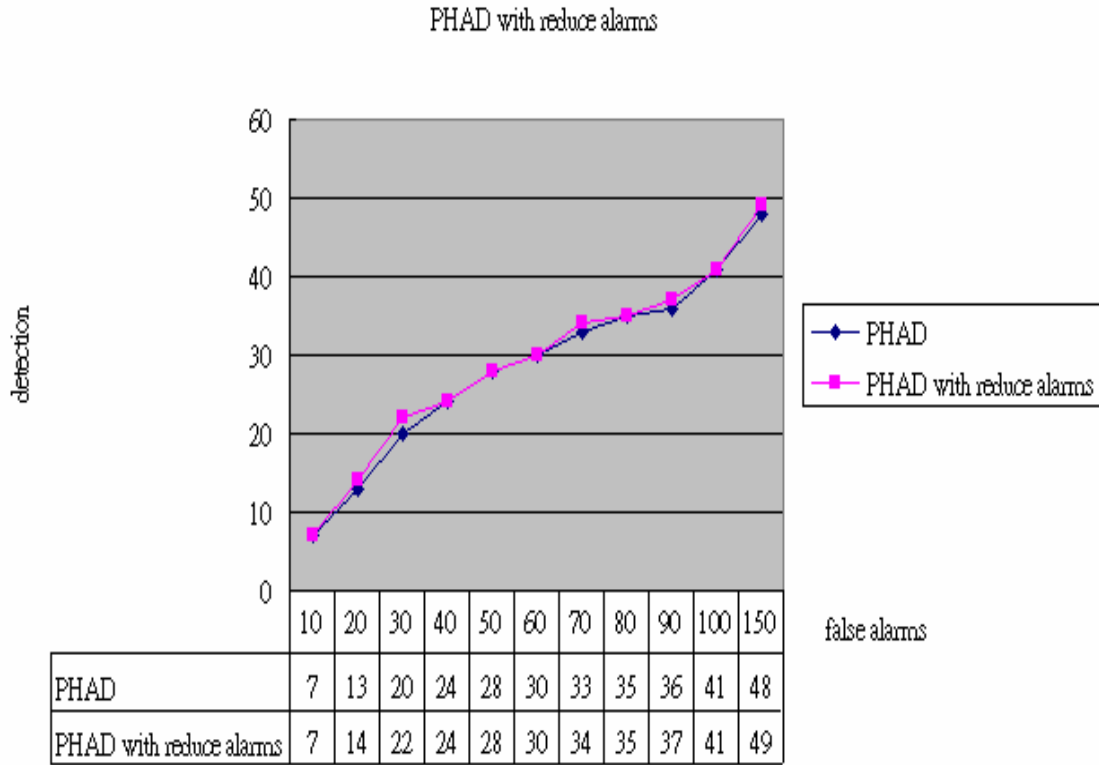


Figure 4.5 PHAD with reduce alarms.

In Figure 4.5, we observe that in each false alarms threshold, the models through reducing alarms has more or equal detection number than original PHAD. In addition, the alarms are reduced from 6260 to 6226.



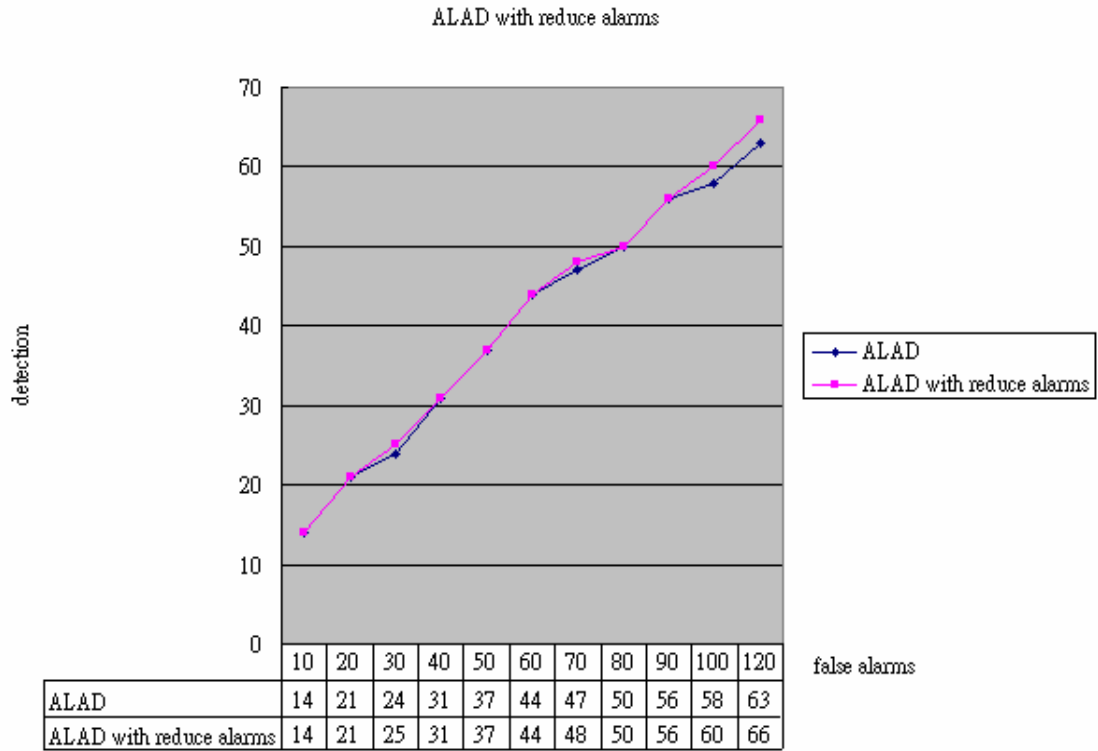


Figure 4.6 ALAD with reduce alarms



In Figure 4.6, we observe that in each false alarms threshold, the models through reducing alarms has more or equal detection number than the original ALAD. In addition, the alarms are reduced from 887 to 821.

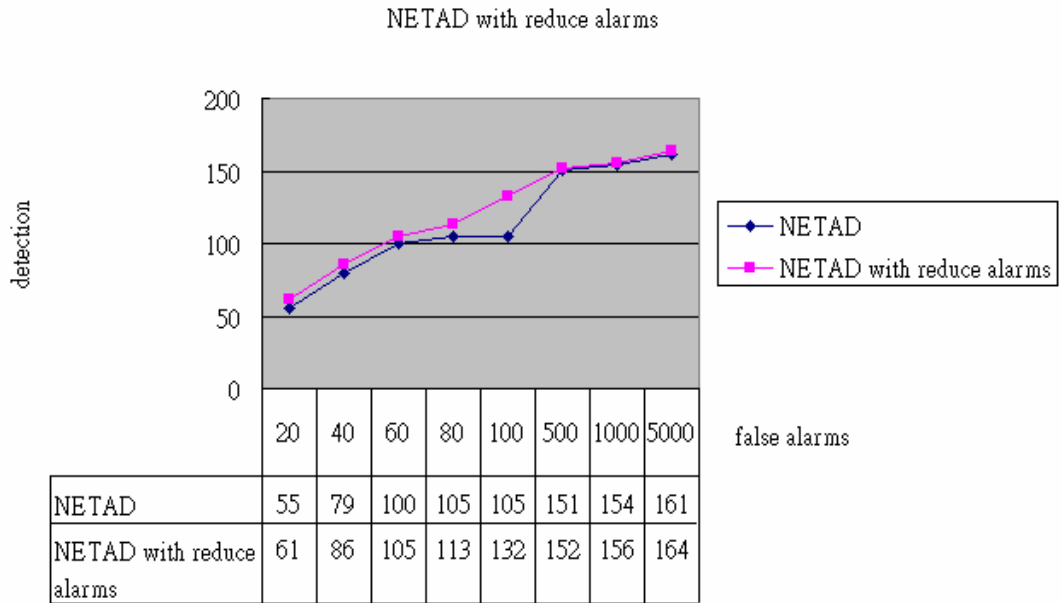


Figure 4.7 NETAD with reduce alarms.

In Figure 4.7, we observe that through reducing alarms, the detection rate is better than the original. In addition, the alarms of the improved NETAD are 23714. It is much smaller than the original, 93753. On one hand, the maximum detection number of NETAD is 169 when the false alarms reaches 8723, while on the other hand the maximum detection number of improved NETAD is also 169 while the false alarms only reaches to 6353.

Another instance, we use the combined model described before, and with reducing alarms.

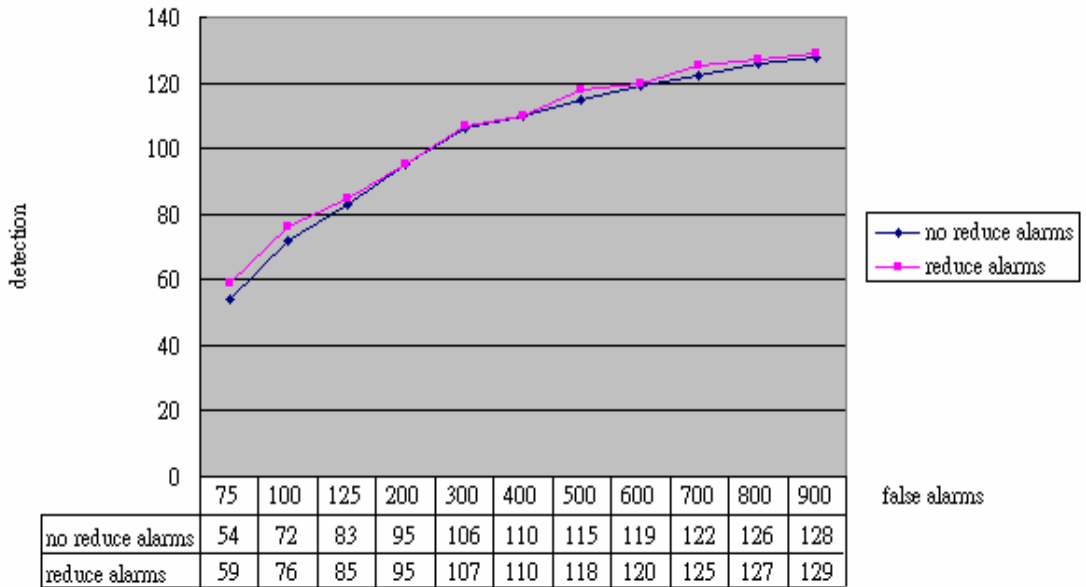


Figure 4.8 merge PHAD and ALAD with reduce alarms.

Figure 4.8 show that reducing alarms also improves the detection rate. In addition, the alarms reduced from 1703 to 1585.

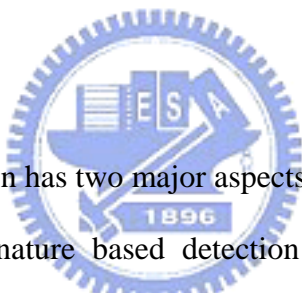
We observe those results via our experiment, we know that the traffic filtering can reduce the input data, and improve the detection. The scheme of reducing alarms can reduce redundant alarms, and leave the necessary alarms for us to detect. It can also improve the detection rate.

# CHAPTER 5

## Conclusion

In this Chapter, we conclude this thesis, and state our work briefly. Finally, we propose our future work.

### 5.1 Conclusion



Popular intrusion detection has two major aspects, one is signature based, and the other is anomaly based. Signature based detection emphasizes pattern matching, however, anomaly based detection system watch for deviations of actual from expected behavior and classifies all ‘abnormal’ activities as malicious. So it needs a training period to establish a ‘normal’ behavior. It can detect novel attack better than signature based system. In our experiment, we use the week 3 of the DARPA-offline intrusion data set for training. Then we use the weeks 4 and 5 for testing. At first, we compare the PHAD and the ALAD models .Then we also exploit the skill of traffic filtering from NETAD, and process it by PHAD. It can not only reduce the input data greatly, but also promote the detection rate. We observe that merge them will generate better performance. Through reducing alarms, we can get better detection rate. Equally, we come at the NETAD models with reducing alarms also has better manifestation.

## 5.2 Future Work

We use the 1999 DARPA-offline intrusion data set in our experiment, but we found a simulation artifact in the TTL field of the IP header which makes the attacks easy to detect, especially in PHAD. So, in our evaluations, we set 0 to this field. In addition, we need a attack-free data for training, and this is not easy for actual network. Therefore, we may use our method with a real data set in the future.



## REFERENCES

- [1] Roesch, Martin, “Snort – Lightweight Intrusion Detection for Networks ”, Proc. USENIX Lisa '99 ,Seattle: NOV. 7-12, 1999.
- [2] Paxson, Vern, “Bro: A System for Detecting Network Intrusion in Real-Time”, Lawrence Berkley National Laboratory Proceedings, 7'th USENIX Security Symposium,Jan. 26-29, 1998 ,San Antonio TX.
- [3] SPADE, Silicon Defense  
<http://www.silicondefense.com/software/spice/>
- [4] Sekar ,R. ,M . Bendre , D.mDhurjati , P. Bollineni , “A Fast Automaton-based Method for Detecting Anomalous Program Behaviors ”. Proceedings of the 2001 IEEE Symposium on Security and Privacy.
- [5] Anderson, D. et. Al., “Detecting unusual program behavior using the statistical component of Next-generation Intrusion Detection Expert System (NIDES) ”, Computer Science Laboratory SRI-CSL 95-06 May 1995.
- [6] Forrest, S., S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, “A Sense of Self for UNIX Processes”. Proceedings of 1996 IEEE Symposium on Computer Security and Privacy.  
<ftp://ftp.cs.unm.edu/pub/forrest/ieee-sp-96-unix.pdf>
- [7] Mahoney, M., P. K. Chan, “PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic”, Florida Tech. technical report 2001-04,  
<http://cs.fit.edu/~tr/>
- [8] Lippmann, R., et al., "The 1999 DARPA Off-Line Intrusion Detection Evaluation", Computer Networks 34(4) 579-595, 2000.

- [9] Mahoney, M., P. K. Chan, “Learning Nonstationary Models for Normal Network Traffic for Detecting Novel Attacks”, Florida Tech. Technical report 2002-08,
- [10] Ghosh, A.K., A. Schwartzbard, M. Schatz, “Learning Program Behavior Profiles for Intrusion Detection ”, Proceedings of the 1st USENIX Workshop on Intrusion Detection and Network Monitoring , April 9-12 ,1999 , Santa Clara, CA . [http://www.cigital.com/~anup/usenix\\_id99.pdf](http://www.cigital.com/~anup/usenix_id99.pdf)
- [11] Mahoney, M., “NETAD: Network Traffic Anomaly Detection Based on Packet Bytes”, 2003 ACM.
- [12] Mahoney, M., P. K. Chan, “Learning Models of Network Traffic for Detecting Novel Attacks”, Florida Tech. Technical report 2002-08,”
- [13] Christopher Kruegel, Thomas Toth and Engin Kirda, “Service Specific Anomaly Detection for Network Intrusion Detection.” April 30, 2002 ACM.
- [14] A.K. Ghosh, J Wanken, and F. Charron. Detecting Anomalous and Unknown Intrusions Against Programs. In Proceedings of the Annual Computer Security Applications Conference (ACSAC’98), pages 259-267, Scottsdale, AZ, December 1998.
- [15] W. Lee, S. Stolfo, and K. Mok. Mining in a Data-flow Environment: Experience in Network Intrusion Detection. In Proceedings of 5<sup>th</sup> ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD ‘99), San Diego, CA, August 1999.
- [16] H. S. Javitz and A. Valdes. The SRI IDES Statistical Anomaly Detector. In Proceedings of the IEEE symposium on Security and Privacy, May 1991.