


私立東海大學

資訊工程與科學研究所  
碩士論文

使用浮水印從事逆向追蹤與結合鑑識科學之  
入侵偵測系統

A Host-based Intrusion Detection System  
using Forensic Techniques with Watermark

Trace Back



指導教授：呂芳懌  
研究生：楊子逸

中華民國九十三年七月

## 中文摘要

全球所建置的網路數目正與日劇增，而在這些持續增加的網路中，許多都是直接或間接的連接上網際網路。凡是與網際網路連線的系統，在資料存取、商業行為以及生產力等方面，都創造了令人振奮的新契機。然而連線也使得該網路面臨了資料被竊取或遭受危害的危險。有心人士會透過作業系統原有的漏洞或網際網路協定本身的缺陷，攻擊網路主機、植入木馬程式、竊取商業機密、篡改資料，甚至阻斷服務，造成企業形象受損或實質上的財務損失。如何有效地確保網路安全，並遏止網路犯罪，已成為刻不容緩的問題。為了加強網路的安全，防火牆及入侵偵測系統的出現正是因應之道。

主機式入侵偵測系統在入侵偵測系統的發展過程當中，扮演著非常重要的腳色。它是藉由檢查作業系統或應用程式所產生出來的稽核檔案來偵測作業系統是否遭到入侵，然而這一種偵測方式通常必須花費相當多的電腦資源，如：處理器的時間及記憶體，及相當長的時間來完成，另一項缺點是無法即時偵測。此外如何將使用者的操作模式及使用行為轉換成數位的資料記錄下來，仍然是一個很難達成的目標。為了改善這些問題，我們提出一個改良式的主機入侵偵測系統，稱為主機式人類行為學入侵偵測系統，Host-based forensic intrusion detection system (HFIDS)。我們也設計了一個中介層，稱為 Intelligent Monitor (IM)，並把它安裝在作業系統的核心及使用者操作介面（例如：Shell）之間，用來蒐集使用者的一切輸入資料。IM 係即時地監測使用者的活動，再藉由統計使用者的習慣，迅速地發現使用者的誤用情形或網路的異常情形。我們不只可以找出那個帳號被盜用，更能找到真正的入侵者。並且 HFIDS 是不需要不斷更新入侵資料庫的，因為它有自我組織及訓練的能力。此外我們更能藉由資料探勘及人類行為學的技术即時地發現協同式攻擊，也就是同時有兩個以上入侵者的搭配攻擊。

逆向追蹤的技术則是本研究的另一個重點，傳統的逆向追蹤技术需要比較

多的系統資源，並且會增加網路的負載，本研究乃是研製一個輕量型入侵偵測器 (Lightweight Intrusion Detector, LID)，除了追蹤由 IM 所發現的入侵攻擊路徑外，更能提供許多有效的資訊給 HFIDS，例如入侵者的攻擊路徑、弱點電腦的分布...等，俾即時(Real-Time)地反應。藉由封包浮水印的技術，在可疑的連線封包中加上一組資訊，以逆向追蹤的方式找出完整的攻擊路徑，讓跳板主機或攻擊主機無所遁形。我們亦採用隱形技術，有效防止駭客攻擊 LID 等。

HFIDS 的主要偵測步驟，包含前處理、分類、特徵化及探勘四個步驟。我們成功地將指令的特性導入頻率域，也就是較常出現的指令給予較低的分數，不常出現及有危險性的指令給予較高的分數，將指令區分成危險、警告及正常三個等級。而前處理階段就是使用 IM 監視使用者鍵入的指令，比較指令的分數及使用者是否曾經使用過該指令，以決定是否讓該指令通過審核而進入作業系統，以降低主機遭受立即性危險的可能性。在 HFIDS 的特徵化階段，除了一邊更新指令的分數外，同時建立及更新使用者的指令集。指令集包含了該使用者曾經使用的歷史指令集，以及他的使用習慣。而分類階段則是藉由偵測使用者的習慣(如：使用指令的習慣)，並對照其他使用者以找出其特徵俾強化偵測率。在探勘階段我們使用資料探勘及統計的技術，同步偵測所有使用者的即時資料，藉以找出協同式攻擊。

**關鍵字：**入侵偵測系統、智慧型監視器、鑑識科學、浮水印、即時偵測

## Abstract

As the rapid growing of network system especially in the quantities and the sizes, those directly or indirectly connecting to the Internet have brought completely new and tremendous opportunities to their owners both on data access, e-commerce as well as productivities. But these connections, on the contrary, may also conduct the threats of data leaking and attacks. By using the vulnerabilities of operating systems or the defects of network protocols, someone may attack the hosts or install a Trojan for stealing business secret, fiddling the data and even issuing denial of services. These activities have seriously hurt the enterprise both in profit and reputation. Currently how to enhance the security to protect a network system and its resources from the threats and attacks has become the most important issue in recent network security researches.

Generally, Host-based detection methods play an important role in developing an Intrusion Detection System (IDS). One of the major concerns of the development is the latency delay. A Host-based IDS system inspecting log files provided by operating systems or applications needs much more time and demand a large number of computer resources, such as CPU time and memory to analyze its log content. Besides, there still a crucial problem of how to quantify human behavior so as risk measurement of a system can be easily performed by simply comparing the quantities derived. In order to improve the problem we propose a forensic-based IDS, called Host-based Forensic Intrusion Detection System (HFIDS), and design a middle layer, called Intelligent Monitor (IM), locating between the operating system kernel and its applications (e.g., shells), in order to gather the inputs submitted by a user for a further analysis. IM on Windows platform is technically implemented by software interrupts (e.g., int 21h), while IM on Linux platform can be implemented with Linux Loadable Kernel Modules. Of course, different platforms have different implementation ways. IM monitors users' activities in a real-time aspect. By defining user profiles previously, it can easily find out the anomalies and malicious accesses instantly. With the help of user profiles, we can not only uncover which account has been misused, but also realize who the true intruder is. There is no need to update the knowledge databases of HFIDS manually. In fact, it is a self-organized and self-training system. Furthermore, we can discover cooperative attacks simultaneously submitted by users as well by using data mining and forensic techniques as the attacks

are performing.

So far, DDoS attacks have focally attracted both research and industry communities' attention. They can be done automatically by using viruses and worms on the Internet. The methods of raising a DDoS attack evolve fast for recent years. Also, it is not easy for a system to find out the intrusion and the infecting paths with traditional security devices, e.g., packet filter, firewall and application proxy, since Viruses, worms and intruders often conceal their identities in the distributed or disguised attacks. This article proposes a framework for tracing the intruding and the infecting paths so that the vulnerabilities on the devices and hosts along the paths can be easily discovered and repaired. As new network device called Lightweight Intrusion Detector (LID) used to figure out these paths quickly is designed and developed. LID traces the intruder in real-time without increasing the network load heavily. It can not only trace the paths of attacks founded by IM, but also provide more information to HFIDS for intrusion analysis.

The working process of HFIDS comprises four phases that are preprocessing, classifying, characterizing and mining phases. The commands of a system are divided into denial, monitor and normal levels ranking by scores. The preprocessing phase is involved to reduce the harm on the systems protected by listening to the keystrokes commands that users have so far typed in, and calculating the commands with the frequency for generating the Commands Score Table (CST). Namely, IM monitors users' activities in a real-time aspect by using the CST. In the characterizing phase, we create profiles for users with Biological and forensic techniques. Lots of attributes are helpful, such as typing habit, the time of a day, typing speeds, commands they have often submitted, commands they have never used, etc. The classifying phase is to recognize and divide them into groups based on their characteristics. In the mining phase, words typed by different users are put together to explore their relationship by involving data mining techniques and statistics techniques. By accumulating scores, we can easily realize who the users having cooperatively initiated an attack are. Are they trying to hurt our system ?

**Keywords** : Intrusion Detection System, Intelligent Monitor, Forensic, Watermark, real-time detection

# INDEX

中文摘要.....	1
Abstract.....	3
Index.....	5
Index of figure.....	7
Index of table.....	8
Chapter1 Introduction.....	9
Chapter 2 History and Motivation.....	14
2.1 History.....	14
2.2 An overview upon intrusion detection.....	16
2.2.1 Data collection.....	17
2.2.2 Detection Techniques.....	18
2.2.3 Response.....	19
2.2.4 Open Issues.....	20
2.3 New technique involved.....	20
Chapter 3 System framework.....	22
3.1 The framework of HFIDS.....	22
3.1.1 User profiles.....	23
3.1.2 Intelligent monitor.....	25
3.1.3 Profile for mining server.....	26
3.2 Lightweight intrusion detector.....	27
3.3 Intrusion scenario.....	30
3.4 Algorithm.....	31
3.4.1 Categorization algorithm.....	32
3.4.2 Sequence algorithm.....	33
3.4.3 Characterization algorithm.....	33
3.4.4 Mining algorithm.....	34
3.4.5 Comparing algorithm.....	35

<b>Chapter 4 The experiments.....</b>	<b>38</b>
<b>4.1 Profile generation.....</b>	<b>38</b>
<b>4.2 User Recognition.....</b>	<b>39</b>
<b>4.3 Recognition trend with number of commands.....</b>	<b>41</b>
<b>4.4 Recognition trend with number of users.....</b>	<b>42</b>
<b>4.5 Recognition rate with specific clusters.....</b>	<b>43</b>
<b>4.6 An advanced test.....</b>	<b>44</b>
<b>Chapter 5 Conclusion.....</b>	<b>45</b>
<b>Reference.....</b>	<b>47</b>

## Index of figure

<b>Fig.1.1 2003 CSI/FBI Computer Crime and Security Survey.....</b>	<b>10</b>
<b>Fig.1.2 Security Technologies Used.....</b>	<b>11</b>
<b>Fig.2.1 Forensic Technology.....</b>	<b>21</b>
<b>Fig.3.1 System framework.....</b>	<b>23</b>
<b>Fig. 3.2 An example of a profile represented by graphs in frequency domain....</b>	<b>26</b>
<b>Fig.3.3 A LIDP Packet.....</b>	<b>28</b>
<b>Fig.3.4 Structure of IP packet.....</b>	<b>28</b>
<b>Fig.3.5 The structure of a watermarked packet in option field of an IP packet..</b>	<b>28</b>
<b>Fig.3.6 A recommended pipeline machine to improve system performance of a LID.....</b>	<b>30</b>
<b>Fig.3.7 Intrusion Scenario.....</b>	<b>30</b>
<b>Fig.3.8 Sequential command sets algorithm.....</b>	<b>33</b>
<b>Fig.3.9 Correspondence of characteristic parameters.....</b>	<b>36</b>
<b>Fig.3.10 Minimum accumulated distance.....</b>	<b>37</b>
<b>Fig.4.1 The trend of recognition rate with the minimum commands for each user.....</b>	<b>41</b>
<b>Fig.4.2 The trend of recognition rate with the growing of users.....</b>	<b>42</b>
<b>Fig.4.5 Comparison with user A and B in Graph.....</b>	<b>44</b>



## Index of table

<b>Table 3.1 Dangerous levels.....</b>	<b>25</b>
<b>Table 3.2 Examples of command ranking.....</b>	<b>25</b>
<b>Table 3.3 LID control message.....</b>	<b>29</b>
<b>Table 3.4 Examples for score table on IM.....</b>	<b>32</b>
<b>Table 4.1 An example of history file on Solaris.....</b>	<b>38</b>
<b>Table 4.2 an example of CP.....</b>	<b>39</b>
<b>Table 4.3 The time required to finish the experiment.....</b>	<b>40</b>
<b>Table 4.4 Specifications of the hardware platform.....</b>	<b>40</b>
<b>Table 4.5 The detail of recognition rate with the min commands for each user..</b>	<b>42</b>
<b>Table 4.6 Recognition rate with CSIE students.....</b>	<b>43</b>
<b>Table 4.7 Recognition rate with Math students.....</b>	<b>44</b>

## 1 Introduction

As having been prosperously and rapidly developed, network technologies recently have brought to us a whole new life and shopping experience. Also, business transactions on the Internet have so far attracted enterprises' and users' attention since it has brought completely new and tremendous opportunities to their owners on data access, e-commerce and productivity to force them becoming invaluable tools so with which enterprises can share information and conduct business from online partners. But security and cyber crimes consideration on the contrary slow down the speed of the market development. Due to the fact that lots of intrusion and attacks causing huge financial losses and data damage comes from the Internet. For example, Feb. 10, 2000 is not a peaceful day for e-companies: eBay, Yahoo, Amazon, CNN and Washington Post had been attacked one by one by Distributed Denial of Service (DDoS). They lost not only the benefit from temporarily stopping their services provided, but also damaged the reliable image established before. According to the survey issued by CSI (Computer Security Institute) and FBI (Federal Bureau of Investigation) [1] in 2003, only 30% of enterprises are willing to report their injuries and only 47% have the ability to quantify their damage. Please refer to Fig. 1.1. Particularly, the total amount of money lost is about 201 million US dollars. The reasons that people are not willing to report their losses are two folds. First, company reputation will be hurt. Second, some of its customers will move to the competitors due to their insufficient confidence. So, hiding the intrusion events is another way to protect the company itself.

As hackers have learned to use the network/system vulnerabilities and imperfection to access private networks and their resources, studies have shown that many organizations have suffered from external and internal network intrusions, especially those resulting in sizable losses of money. When penetrating systems by taking advantage of bugs or by acquiring passwords, hackers can not only raise the Denial of Service (DoS) and fiddle the homepages, but also steal some sensible data, such as the commerce secrete and military deployment. Both of them can cause the unimaginable consequence, e.g., the patent violation, the raises of a war and even the world war.

<b>The Cost of Computer Crime</b>		In 2003, 75% of our survey respondents acknowledged financial losses, but only 47% could quantify the losses.											
The following table shows the aggregate cost of computer crimes and security breaches over a 48-month period													
<b>How Money Was Lost</b>													
	<b>Lowest Reported</b>				<b>Highest Reported</b>				<b>Total Annual Losses</b>				
	00	01	02	03	00	01	02	03	00	01	02	03	
Theft of proprietary info.	\$1K	\$100	\$1K	\$2K	\$25M	\$50M	\$50M	\$35M	\$66,708,000	\$151,230,100	\$170,827,000	70,195,900	
Sabotage of data of networks	1K	100	1K	500	15M	3M	10M	2M	27,148,000	5,183,100	15,134,000	5,148,500	
Telecom eavesdropping	200	1K	5K	1K	500K	500K	5M	50K	991,200	886,000	346,000	76,000	
System penetration by outsider	1K	100	1K	100	5M	10M	5M	1M	7,104,000	19,066,600	13,055,000	2,754,400	
Insider abuse of Net access	240	100	1K	100	15M	10M	10M	6M	27,984,740	35,001,650	50,099,000	11,767,200	
Financial fraud	500	500	1K	1K	21M	40M	50M	4M	55,996,000	92,935,500	115,753,000	10,186,400	
Denial of service	1K	100	1K	500	5M	2M	50M	60M	8,247,500	4,283,600	18,370,500	65,643,300	
Virus	100	100	1K	40	10M	20M	9M	6M	29,171,700	45,288,150	49,979,000	27,382,340	
Unauthorized insider access	1K	1K	2K	100	20M	5M	1.5M	100K	22,554,500	6,064,000	4,503,000	406,300	
Telecom fraud	1K	500	1K	100	3M	8M	100K	250K	4,028,000	9,041,000	6,015,000	701,500	
Active wiretapping	5M	0	0	5K	5M	0	0	700K	5,000,000	0	0	705,000	
Laptop theft	500	1K	1K	2400	1.2M	2M	5M	2M	10,404,300	8,849,000	11,766,500	6,830,500	
<b>Total Annual Losses</b>									265,337,990	377,828,700	455,848,000	201,797,340	

CSI/FBI 2003 Computer Crime and Security Survey  
Source: Computer Security Institute

Fig.1.1 2003 CSI/FBI Computer Crime and Security Survey

Encrypted Login, Firewalls, Reusable Passwords, Anti-virus Software, Encrypted Files, Biometrics and Access Control, etc, see Fig.1.2. So far many organizations have forced to protect their systems and networks from infection, intrusion and attacks by using anti-virus software and firewalls. Viruses are some special programs written for infecting a system in order to destroy it or make the system work abnormally. They are often pretty small in size and can be a piece of assembly language, C language, VBscript code, or even a standalone program. Some viruses have the ability to clone themselves and infect other normal programs. Their objective is to play a trick with their end users, stealing their private and secret information or even destroying the operation systems to paralyze the computers. The main functions of Anti-virus software are to scan and remove these special programs or just isolate the programs that have been infected.

As one of the most popular devices/applications for solving network security problems, a firewall protects its network and the hosts in the network by checking some of the fields in the packets that go through it to see if the packets are legal or not. An illegal packet will be discarded and its network connection will be disconnected. The fields checked include the source IP, destination IP, Type of services, ports...etc.

The administrators have to decide what kinds of services we may have, and what kinds of connections whose IP should be checked. We call these activities and behavior that a firewall is requested to do the security policy.

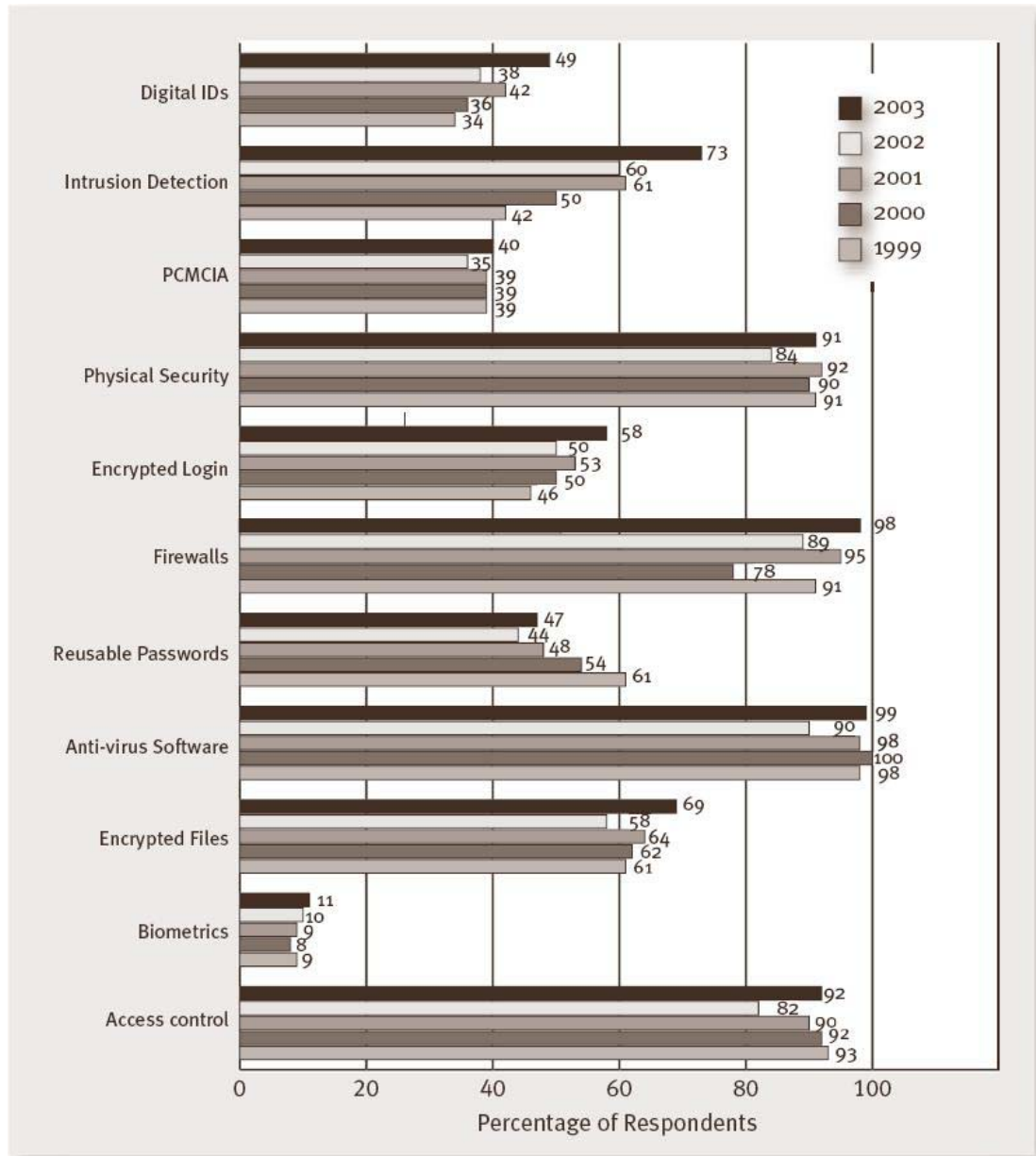


Fig.1.2 Security Technologies Used

The term physical security, though, is arguably broad. Some people interpret it as simply asking whether the office premises as a whole are locked after hours or not, while others may quite justifiably interpret it as if there are some specific facilities (e.g., special alarms or locked areas) designed to protect computers and network assets or not.

Generally, access control is a well-known protection mechanism and is often implemented by checking password and user ID both of which usually are alphanumeric data and often are not long enough to protect the systems safely. Some principles of evaluating if the generation and the usage of a password is good enough or not have been raised [2]. First, the password must be long enough. More than eight characters are recommended. Second, it must be a combination of digits and letters. Third, if possible avoid using name, birthday, phone number...etc., as the passwords. Finally, change the password frequently.

On the other hand, intrusion detection systems (IDS) acting like a firewall have grown rapidly in recent years. Both of the two devices can be implemented either by software or hardware. They play almost the same role, detecting unusual activities and intuiting alarms. But often IDS is cleverer in that firewall can only follow the policies that the administrators submit, and then block the illegal connections. However, an IDS rejects or terminates these connections according to what the users have done. For example, an ordinary user is not allowed to delete any system file. However, if one user successfully connects to the host by using someone's legal account, i.e., the super user's account, firewall will do nothing even he/she tries to delete a system file. But an IDS analyzes the users' behaviors to judge their illegalities. When some suspicion is discovered, it can terminate the connection and issue an alarm to the administrators.

Imagine that there is a strange person standing in front of our house, looking around, investigating the surroundings, then approaching the front door and trying to turn the door handle. However, the door is locked. He moves to the nearby window and gently tries to open it. It is locked, too. It seems that our house is secure. So why do we install a guarding system in the house? The similar question is often raised. If firewalls are already established, operating systems have been immediately patched once patching programs are available, and passwords have been properly checked, why do we redundantly set up an IDS? The answer is simple: because intrusions do not disappear. Just as people sometimes, for example, forget to lock a window, the administrators occasionally forget to update the rule sets of the firewalls administrated or due to some reasons postpone the update.

Even with the most advanced protection systems, no one dares to say the

network and the system being protected is one hundred percent secure. In fact, most computer security experts agree that, as providing user-desired features such as network connectivity, we'll never achieve a completely secure system. According to the conclusion concluded above, clever intrusion detection techniques and systems for discovering and reacting computer attacks and intrusions need to be urgently developed.

In this article, we bring up a new IDS, called Host-based Forensic Intrusion Detection System (HFIDS), which is developed with Forensics, Biometrics, data mining and watermark techniques. We use the Forensic technique to profile the user behavior in order to automate the maintenance of user profiles, data mining technique to find out the cooperative attack, and watermark technique to trace back the hackers or intruders. The goal of our system is to detect the intrusion real-time, effectively and efficiently.

The rest of this article is organized as follows: Chapter 2 surveys the IDS history and the approaches used. Chapter 3 brings up the framework of the improved IDS. In Chapter 4, we explain the steps, algorithms and some formulas for detecting hackers. Chapter 5 concludes and state the future works.

## 2 History and Motivation

### 2.1 History

At the very beginning, the administrators of a system detect intrusions by sitting in front of the console monitoring user activities to see if their system works in a legal and normal status or not. Here, we give two abnormal examples. A user being going on his/her vacation locally logs in the system or a seldom-used service unusually becomes active. Although effective enough at that time, this early approach was special and infeasible since the administrators can not go away or out focus from the console at all the time.

The next stage of the detection process, from late '70s to early '80, was to inspect the logs. System administrators traditionally printed the audit logs on paper, which were often stacked up to four to five feet high by the end of a week, and then search the evidence of hacker behavior, an unusual and/or malicious activity, in such a stack. It was obviously very time-consuming. With this overabundance of information and manual analysis, the administrators mainly used the log content as a forensic data to identify the cause of a particular security incident after the incident happened. James P. Anderson [3] was the first one proposing the guidelines of transforming the requirements of auditing data into documents automatically. He wrote a research plan in 1980 for U.S. air force and announced the "Reference Monitor (RF)", which helped a lot in developing intrusion detection techniques in the past years. The inspection upon the logs was first carried out. He also suggested that we should transform the data collected, by the auditing mechanism, into useful information to reduce the redundant and unnecessary security audit data before the administrators start tracing the hacker behaviors.

As storage media becomes cheaper, currently logs are audited and researchers also start to develop programs to analyze the log data. An early abstract model of typical IDS was provided by Dorothy Denning and Peter Neumann and finally adopted by the research plan, supported by "Space and Naval Warfare Systems Command (SPAWARS)" of the U.S. navy, to realize the prototype of a real-time intrusion detection system, named "Intrusion Detection Expert System (IDES)" [4-6]. The working principle of the IDES was to define and uncover the relationship

between an abnormal behavior and its corresponding misuse based on previously established profiles, a data structure involving a statistical matrix and model to describe the behavior among system objects. Activity rules are specified in the profile when the behaviors are legally allowed. The abnormal behavior in the project was defined as the activities that are rare and unusual and essential portion in usage from statistics viewpoint. The definition was widely used as the fundamental of intrusion detection researches in 1980s. In 1987 Denning announced a paper which was considered as the milestone in the area of developing intrusion detection systems [7].

Besides, IDES uses a kind of hybrid skeleton, an integration of an anomaly detector and an expert system to assist the detection of hacker behavior. The former uses the statistics skills to characterize the abnormal behaviors, while the latter invokes the rule-based approach to detect the known violating activities and is capable of helping the anomaly detector to reduce the system risks since an intruder with high patience can change his/her actions smoothly in order to hide the intrusion activities from the discovery by the anomaly detector. However, the expert system can fit up the drawback.

After the announcement of the IDES, a lot of prototypes of IDS have been developed, such as Audit Analysis Project [8] which analyzes the UNIX shell data, Discovery expert system [9] which is designed special for use of a database system, Haystack [10] which detects the features, (ex: the period of time of every session, the number of file being opened, CPU resources used...etc) of an abnormal behavior, Multics Intrusion Detection and Alerting System (MIDAS) [11] which uses its system information to expand the traditional audit data into session profiles, Network Audit Director and Intrusion Reporter (NADIR) [12] which is the first network-based intrusion detection system (NIDS), The Network System Monitor (NSM) [13] which invokes the matrix-based methodology to analyze the hacker behavior, Wisdom and Sense [14] which chooses statistics and rule-based approach to analyze log data and Distribution Intrusion Detection System (DIDS) [15] which is the first distributed system integrating the Host-based IDS (HIDS) and NIDS. However, most of their analyses are comparatively slow and computationally intensive. Therefore, intrusion detection programs are usually run at midnight when the systems' loads are light. In other words most intrusions are detected afterward.



In the early '90s, researchers had developed real-time intrusion detection systems that inspected audit data as produced. That is, they detected attacks and attempting attacks immediately enabling the real-time response, and, in some cases, attacks prevention.

Recent intrusion detection efforts have centered on the development of the products that can be effectively deployed in a large network. There is no easy task. If we attempt to increase security concerns, encounter new detecting techniques, and compromise with the continuous changes of the surrounding computing environments, much more efforts are needed.

## **2.2 An overview upon Intrusion detection**

The key task of intrusion detection is seemingly simple, i.e., detecting intrusions. However, the task is difficult since an IDS does not detect intrusions directly. What it does is identifying the evidence of an intrusion either when the intrusion is in progress or at some time after the intrusion is completely performed.

Such evidence is sometimes considered as an attack's "manifestation." If there is no manifestation, if the manifestation supports insufficient information, or if the information is untrustworthy, then the intrusion is not able to be detected. For example, suppose a house monitoring system is analyzing a camera picture showing that a person is standing quietly in front of our door. The picture is the manifestation of an intrusion. However, if the camera lens is dirty or out of focus, the system will be unable to judge whether the person is a burglar or the owner of the house.

Generally, there are two types of IDSs: NIDS and HIDS. An HIDS (e.g. MIDAS [11]) as stated above uncovers attacks by investigating system events recorded in log files precisely. So fewer false alarms will be generated and more types of intrusions can be detected. Its drawback is that illegal behaviors can be discovered only after attacks are performed, not instantly. An HIDS often consumes a large amount of computer resources, such as CPU time and memory. On the contrary, an NIDS (e.g., NADIR [12]) can find out attacks immediately and response appropriately. Its key drawback is the detecting coverage. What it can detect is usually the abnormal behaviors below the session layer of the OSI Model. But an application Proxy which

can analyze activities of the application layer is an exception. NIDS sometimes also slows down the network transmission speed and consumes part of network bandwidth.

An application Proxy, providing a broader view than a NIDS by inspecting the data of each connection, is basically an enhanced NIDS. It uses content-aware technique to detect the attacks or intrusions carefully. It can also trace the intrusion paths in simple ways (e.g. audit the source IP and destination IP). But as the network environments become much more complex, the efficiency will be lower. In this paper, we improve this weakness by using watermarked packets and sniffing techniques, lightly scarifying the load of existent networks, to explore the true intrusion paths and point out the vulnerable network devices, hosts and applications located along the paths.

### **2.2.1 Data Collection**

In order to accurately detect network intrusions, enough reliable data concerning the activities of the target system is needed. However, collecting reliable data is a complex issue due to the fact that an operating system, a router and a firewall often provide some forms of audit in their logs for different kinds of users. The contents are frequently limited to either the security-relevant events, such as some failed login attempts, or a complete report of the system calls invoked by all the processes. The information contained in these logs is often very simple, such as the beginning and the end of a network connection, or a complete record of the packets that go through the system.

The amount of information that a system collects for its system activities is a trade-off between the overhead and the effectiveness. An information system that records its system events in detail may have substantially degraded its performance and consumed enormous disk storage. For example, a complete log of 100-Mbit network with Ethernet links demands hundreds of Gbytes of disk space for its daily traffic.

Collecting information is expensive. Collecting the right information is important. Determining what information should be logged and where the information can be collected is an open problem. An example is that monitoring the ID of the user

to record who logs in some workstation does not help the detection of intruder, i.e., stepping stone and masquerade. On the other hand, if the audit data includes the content of users input, monitoring the users' ID may be reasonable.

### **2.2.2 Detection Techniques**

There are two main kinds of intrusion detection techniques that are anomaly detection and misuse detection.

Anomaly detection using profiles to gather the normal behaviors for its users and/or applications are performed under the assumption that the instructions/commands that intruders send are different from those recorded in the profiles [16]. For example, we profile certain user activities quite precisely. Suppose a particular user routinely logs in the system around 8 a.m., reads the mails by invoking PINE, writes documents with VI, and has very few file access errors...etc. If the system notices that the same user logs in at 1 a.m., starts using compilers and debugging tools, and has numerous file access errors, we can conclude that suspiciously there is a threat upon the system.

The main advantage is that, by defining what "normal" is, they can discriminate whether a violation is a part of the threat model or not, even a previously unknown attack. However, the payment of detecting an unknown attack is bringing up a high false-alarm rate. Also, the training process in a highly dynamic environment is hard to accurately perform.

Contradictorily, a misuse detection system (MDS) essentially defines what the wrong is in a database, containing attack/intrusion patterns/signatures, and compares them with the current audit data to look for the evidence of the corresponding attacks [17] [18]. For example, if an ordinary user uses a special URL link to access some websites developed on the IIS<sup>Microsoft</sup> [19] and executes a privileged application (e.g., CMD.EXE) to do something with no limitation [20], exploiting the lack of file access checks, an MDS can detect such an intrusion.

The main advantage of an MDS is that it focuses the analysis of the audit data and typically produces few false alarms. Their main disadvantage is only capable of

detecting known attacks with well-defined signatures. As a new attack is discovered, the developers have to manually model its features and add them to the pattern/signature database.

We can also phenomenally classify IDSs based on what they behave. An NIDS inspects packets entering or coming out from a network segment, typically the one serving an enterprise or the major portion of an intranet, and is able to simultaneously monitor several hosts. However, such an NIDS often suffers poor performance, especially with increasing network speeds. Many NIDSs make pathologies as a packet is fragmented and suffer from resource exhaustion problems when they must maintain the information of the attacks for many hosts attacked over a long period of time. In spite of these deficiencies, they are still popular because they not only are easily to be deployed and managed as standalone components, but also have little or even no impact on the performance of the system protected.

An HIDS operates on the host protected, inspecting the audit or log data to detect intrusion activities. A variety of log and audit functions can serve to drive IDS algorithms; these can be supplemented by sensors that monitor the interaction of applications with the host operating system. An HIDS can monitor specific applications in the ways that are difficult or impossible to be performed for an NIDS and then detect intrusion activities without creating externally observable behavior. However, they substantially affect performance. Also, successful intrusions that gain high levels of privilege might be able to disable HIDSs and remove the tracks of their operations. An intrusion that installs UNIX root kits is a typical example.

### **2.2.3 Response**

When detecting an intrusion, an IDS responds to the system administrator the degree of danger with a previously defined form which is different in different IDS due to the fact that some IDSs provide more aggressive information, such as paging the administrator, giving an alarm siren, or even raising a counterattack, than others. Of course, some attributes including an alert and a description of the intrusion detected are commonly in use.

A counterattack may send back messages to the hacker, notifying his/her ISP,

and/or reconfiguring the router of our network to block the packets received from the hacker by checking if the hacker's IP address is the source IP of the packet. However, an aggressive response may be dangerous due to the launch of the messages against the innocent victim V if the hacker's IP is spoofed. For example, a hacker attacks a network using a spoofed connection, a connection with the victims IP as the source IP. If the IDS detects the attack and blocks the traffic from that spoofed address, it would effectively be a DoS attack to prohibit the service provided by V.

#### **2.2.4 Open Issues**

Although intrusion detection techniques have evolved rapidly in the past few years, many important issues remain unchanged. First, an IDS must be effectively enough to detect a wider range of attacks with fewer false alarms. Second, an IDS must keep pace with the development of the modern networks in the aspects of increasing sizes, speeds, and dynamics. Finally, we need much more analysis techniques especially those capable of identifying the attacks against the whole network. In this article, we try to solve this issues by involved some new techniques. Some techniques were the first time that been used to intrusion detection researches, and the others were the first time that been proposed.

#### **2.3 New Technique Involved**

Recently, more and more techniques are involved in the development of intrusion detection system such as HMM (Hidden Markov Model) [21], Feedback control [22], Fuzzy [23] and Data mining [24]. Most of them are based on the statistics theory. The most particular one is the forensic technique (i.e. Biometrics) [25]. Forensic science is any science used for legislation or lawfulness purposes, and therefore provides impartial scientific evidence for use in the courts of law and in a criminal investigation and trial. Forensic science is a multidisciplinary subject, drawing principally not only from chemistry and biology, but also from physics, geology, psychology, social science, etc. Examples of forensic science applications include face recognition, fingerprint analysis and handwriting recognition helping the identifying of human being (See Fig.2.1).

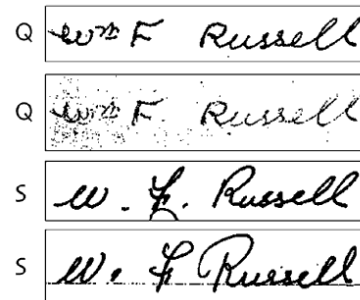
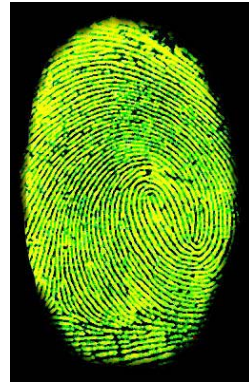
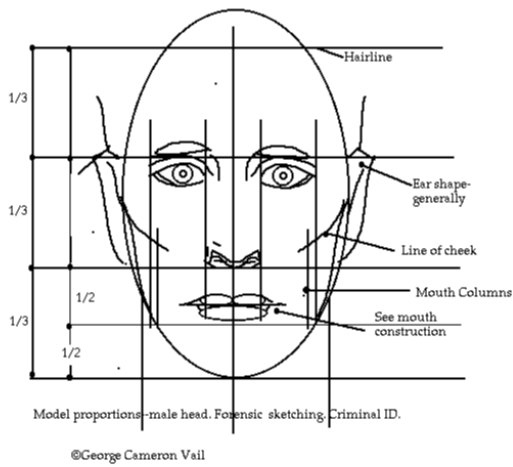


Fig.2.1 Forensic Technology

In this article, we invoke Forensic technique to identify human behaviors, such as the typing habit, typing speed, error rates and the commands/applications used. We profile the user operation behaviors and quantize them into numbers in order to analyze them detailedly. We will describe more about it in the next chapter.

### **3 System framework**

In order to overcome the defects of traditional IDSs, we propose an improved architecture by using host-based, forensic and data mining techniques to detect intrusions. Also we propose a tracing method to enhance the system by uncovering the intrusion paths, so that the stepping stones and the hosts with vulnerabilities or network protocol detects can be easily identified.

#### **3.1 The Framework of HFIDS**

IM, a middle layer subsystem, locating between the kernel of an operating system and its applications (e.g., shells, graphics user interfaces), is designed to gather inputs and activities that a user submits. It can not only reveal which account has been penetrated, but also prevent the system protected in real time from an immediate damage initiated by the intruders (See Fig.3.1).

The role of PMS module is to collect the data and commands what a user has so far entered as the user's profile and then compute the statistic data relationships among the profile data. So that we can find out the user's behaviors and carry out his/her forensic heuristics. That is, user profiles are generated and developed to easily and instantly uncover the abnormal and malicious accesses. Furthermore, we can discover cooperative attacks submitted by several users with data mining techniques. Actually, two kinds of profiles are used in this research: commands set profile (CSP) and Characteristics Profile (CP). A CSP collecting the command sets that a user has ever submitted is used by IM to detect the immediately attacks. A CP consisting of the users' features including the typing behavior extracted from his/her CSP can help PMS to detect the intruders.

LID is a virtual device because it can be installed in an operating system, in a router and or even a standalone network device. It can be widely deployed to trace the intrusion behavior from network viewpoint. The main functions of LID have two folds: mixing the suspicious packets with a special watermark and tracing the intrusion paths by checking the routes that watermarked packets go through.

### 3.1.1 User Profiles

User profiles, widely used in different domains, can be implemented with different data structures. But their purposes, describing the objects of a system or the behaviors of a user, are the same. A NIDS of the early stage used profiles to record network activities, such as bit-rates, session information, packet types and protocols used and so on. Profiles in a HIDS are invoked to keep the information of the system resources, such as the statuses of CPU usage and memory allocation and the information of a system crash.

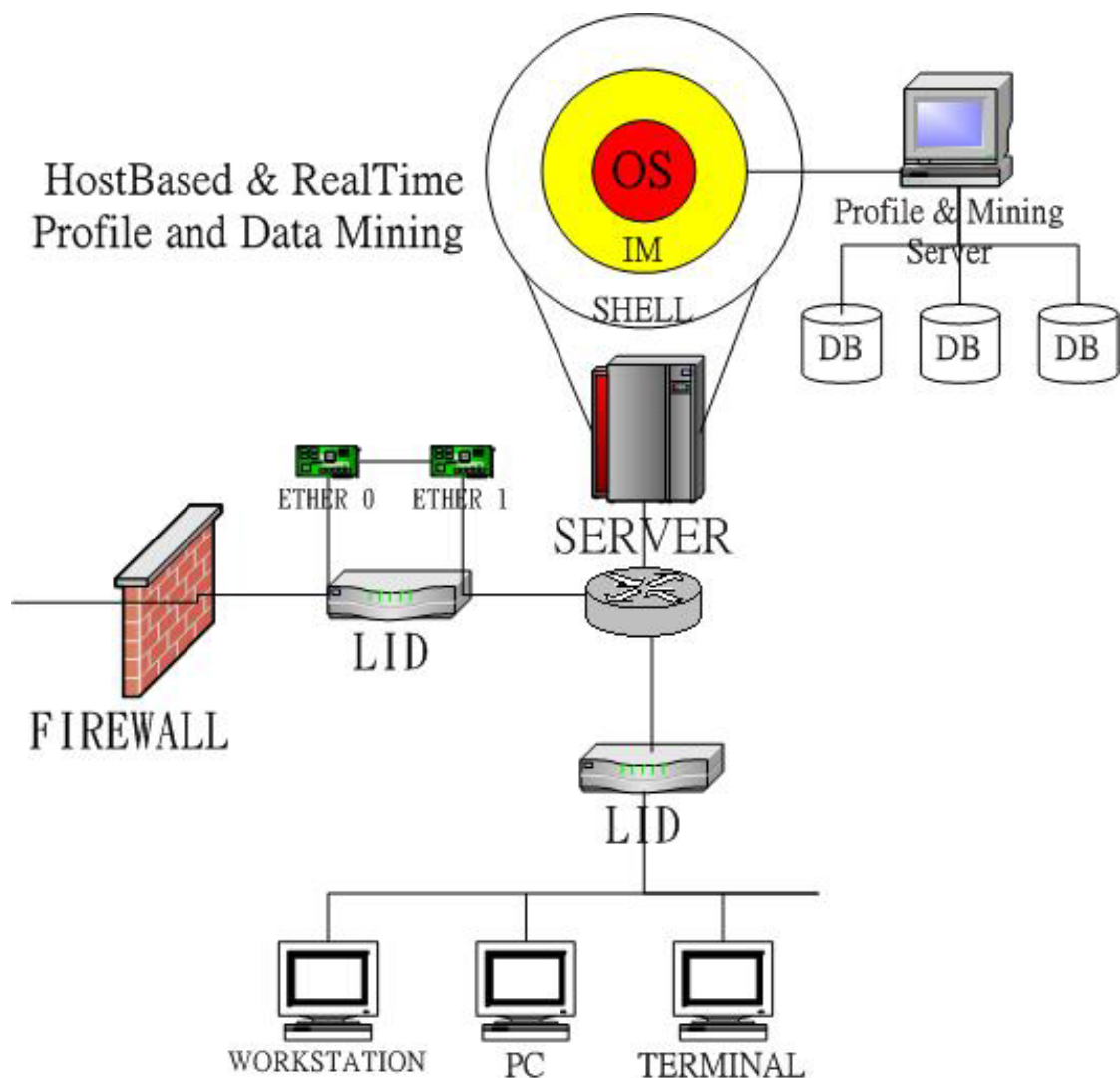


Fig. 3.1 System framework

In forensic medicine, the forensics are defined as the things, such as teeth, DNA sequence and et al, that can uniquely identify a person. We design a profile technique to represent the features of a person, such as what commands they have typed, what



programs they have run and even what habits they have ever performed, and invoke PMS to uncover the user's habits, such as John likes to enter the Internet, Mary enjoys writing articles and Joe is a software engineer. So user behaviors in detail, such as his/her typing rates, error rates, intrusion models and sequence patterns of commands, can be clearly identified.

In our early research, we kept all the words a user has typed. The size of a profile then increased rapidly and it finally became unmanageable and not maintainable. However, Y. Okazaki, I. Sato and S. Goto [26] raise a way to transform system calls into numerical data, and we got some idea on it [27]. In the following, we state the way to create CSP and CP and their main function in detail.

We classify commands into different levels and quantify commands with values (see Table 3.1). One with a smaller value shows that the command appears frequently and is less dangerous, while others with larger values represent that they rarely occur and may be dangerous. We rank each command  $C$  with a three-stage process. First, calculate the frequency that  $C$  has been submitted. Second, refer to the frequency of each command and then assign  $C$  a value (ex. Table 3.2). Finally, replace  $C$  with some relative high scores if it is a dangerous command. We now give each user  $U$  an average command score (ACS) which is derived from his/her own profile, indicating if  $U$  is dangerous or not. We categorize the commands into three levels: normal, monitor and denial. Commands of Normal mode are those frequently used by almost all the users. So we give them the scores lower than 50. It is fine for users running those commands (e.g., cp). Commands of Monitor mode are those invoked by a few people and are in an fuzzy situation. These commands are usually deployed by engineers, programmers and super-user of the system. We delay the commands until HFIDS finishes its security checking by comparing them with the user's CSP. Commands belonging to denial mode are those rarely used ones. Only the administrator of the system invokes them to rebuild or maintain (e.g., kill) the system. In this article, a command may be either a shell command of user level or a system call of system level. If one should be addressed and emphasized, we will particularly describe it.

Table 3.1 Dangerous levels

Level	Score
Normal	$1 < S < 50$
Monitor	$50 < S < 100$
Denial	$S > 100$

Table 3.2 Examples of command ranking

command	score	Command	Score
cp	10	Adduser	78
mkdir	5	Cd	2
kill	101	Mv	12
ps	56	telnet	25
su	95	Chmod	48

### 3.1.2 Intelligent Monitor

In order to improve the weakness and latency delay of an HIDS, IM monitors users' activities in a real-time aspect and gathers users' inputs for further analysis. IM on Windows platform is technically implemented by software interrupts (e.g., int 21h) and is loaded with booting batch files. On Linux platform, IM also is implemented with software interrupts (e.g., int 03h) but loaded with Linux Loadable Kernel Modules. Of course, different platforms may have different implementation ways and loading time.

IM has three working phases. In phase 1, it ranks each command  $C$  that a user enters by looking up CST. If  $C$  is privileged ( $S > 50$ ), it will be temporarily held until the second phases finished for preventing the host computer from an immediately attack. CST can be pre-estimated from history files or periodically calculated from online data. Once a user  $\mu$  submits at least one privilege command,  $\mu$ 's score should be in its monitor level, IM will move  $\mu$  to its next phase. In phase 2, IM compares the commands that  $\mu$  entered, say UCS, with those in the U's CSP (UCSP in short) to see whether UCS matches some of the command set in the UCSP. If yes, IM passes UCS to the operating system kernel. If not, UCS will be held up, and a

logout command will be submitted if necessary. For example, UCS are the commands/applications, such as KILL, FORMAT and FDISK, that will immediately hurt the system. This phase may reduce false alarms due to the checking of UCSP. Phase 3 updates  $\mu$ 's profile (both CP and CSP) and requests PMS to re-calculate the statistics for a future identification of user behaviors and habits. The purpose of this phase is to realize the ideas derived from forensic viewpoint and to establish a knowledge accumulation feedback mechanism.

### 3.1.3 Profile for Mining Server

After collecting the online data, we can transform the data into the frequency domain as a graph  $G$  based on the command sequence currently entered for human visualization. Fig. 3.2 gives an example. Of course, we can compare  $G$  with the set of graphs  $GS$  transformed from the users' profiles (CP + CSP) by shifting  $G$  in each element of the  $GS$ , so that the possible user who submits the UCS can then be easily identified. Fig. 3.2 shows that the current user may be an intruder. We will discuss more details about it in Chapter 3.4.5.

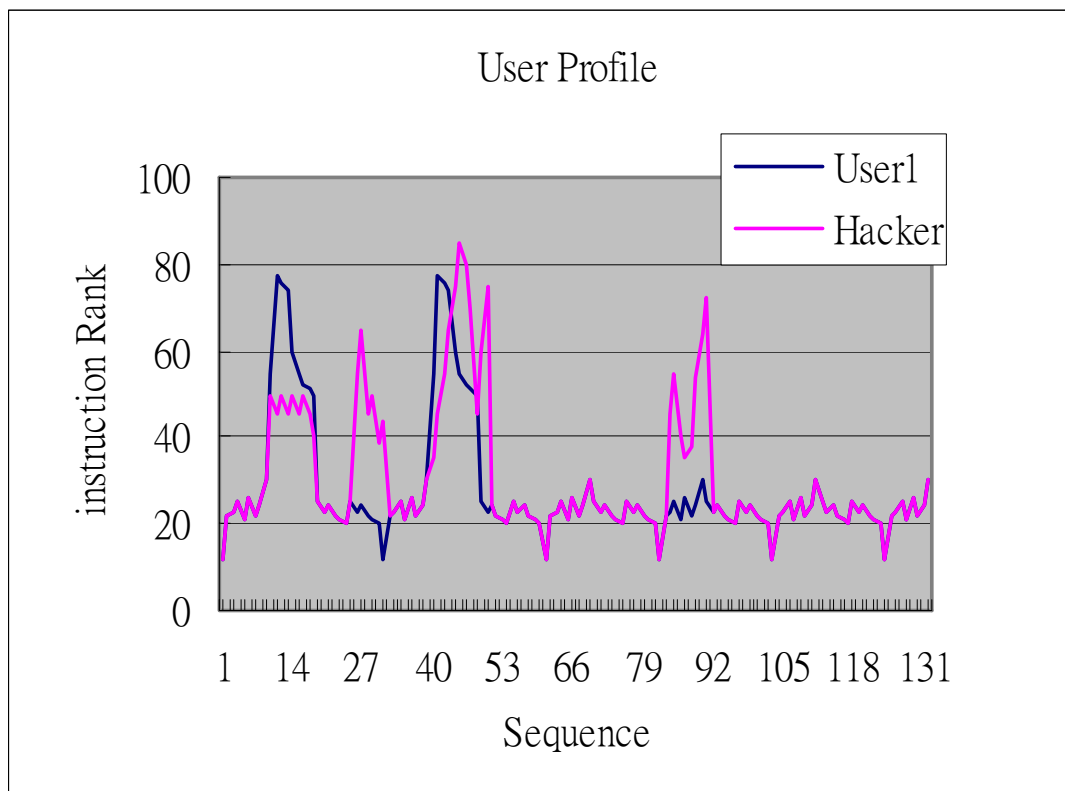


Fig.3.2 An example of a profile represented by graphs in frequency domain

Here we propose some ideas to explore and mine one user's behaviors and habits from his/her history data by identifying the specific features which is often the best way to differentiate a user from others. A feature often consists of several commands with their execution sequence.

In an enterprise, people of different positions often play different roles and do different jobs. But the same position, such as a secretary, often does the same or the similar things. In this article we propose a mining algorithm to find out the differences. We partition a user's history file into fragments each of them contains the commands submitted within the period of time from his/her login to the corresponding logout. With these fragments we can reconstruct the user's behaviors. For example, everyday at the beginning of the office hours, secretary A likes to receive mails (e.g., pine, elm) first. Thereafter all day long she runs the program periodically, once for every thirty minutes. Secretary B likes to enter the Internet first by the browser Netscape [28] to collect orders for goods and then receives mails from web-mail servers. Now their forensic features can be carried out by the PMS. We will discuss more details about it in Chapter 4.

### **3.2 Lightweight Intrusion Detector**

LID is an intrusion tracer located somewhere in a network, such as on the routers, server hosts and even the gateways, to help the network protection. LID does not use the standard transmission mechanism, since it does not provide any port service. Also we turn on the network adapter's Promiscuous Mode for LID in order to receive all the packets that flow through. That is why LIDs are invisible but still capable of tracing intrusion paths. Unlike the traditional NIDSs tracing the anomalous connections with their own working resources (e.g., CPU time, memory), we use "Watermark" [29] to silently find out the intrusion paths and avoid heavily increasing network load. A LID is always in its sleepy mode while nothing happens. When one IDS of the network system discovers that there is an intrusion, it wakes up the LIDs that the nearest by using LIDP to request a tracing service. Then the nearest LID becomes the Mater LID (MLID) which is in charge of mixing the specific passing packets with watermarks. All the LID will monitor the packets that flow through them to see if there is a watermark or not. Namely, all the connection paths of an intrusion or several intrusions will be monitored simultaneously. In order to hide the LIDs, we

use LIDP (LID Protocol) instead of general transmission mechanism for an indirect communication. A LIDP packet is constructed bytes data appended to the normal ICMP packet to carry the control messages of a trace. Please refer to Fig. 3.3.

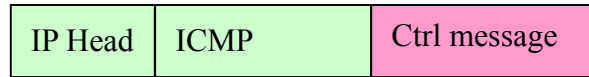


Fig.3.3 A LIDP Packet

**Watermark Packets:** Currently, most of the network applications and Internet services use TCP and UDP, conveyed by IP protocol, as the transport layer network protocols. Figure 3.4 shows the structure of an IP packet. We hide the watermarks in the “options” field which is 32bits in length. So there are up to a total of  $2^{32}$  possible watermarks, each representing a network connection.

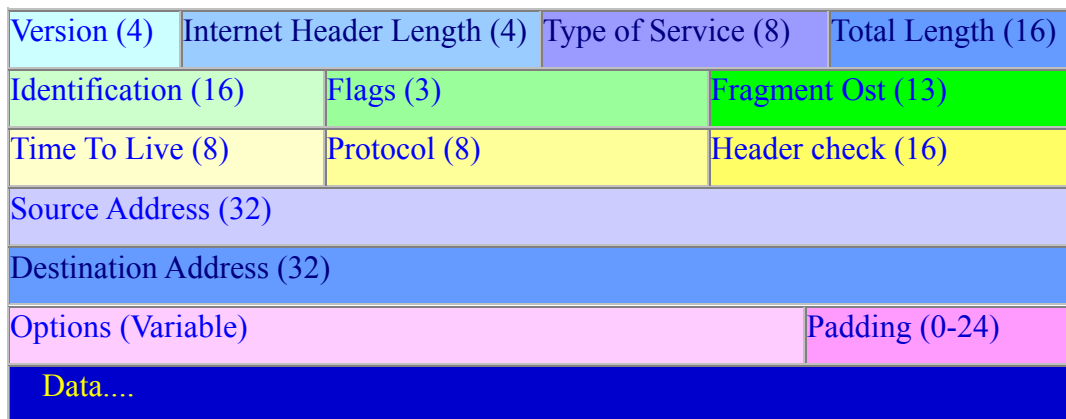


Fig.3.4 Structure of IP packet

A watermark consisting of two fields, LIDid (LID’s identifier) and serial numbers, as illustrated in Fig.3.5 is designed to avoid duplicated ambiguous. The LIDid is the identity of an LID, while the serial number is generated to distinguish different connections.

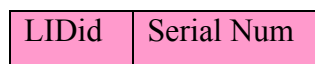


Fig.3.5 The structure of a watermarked packet in option field of an IP packet

**LID:** A LID has two main functions:

1. Checking watermarks: if yes, the LID records the watermark and the routing paths, consisting of the source and destination IPs and MACs of the connection.
2. Checking control messages: there are two types of control messages the tracing command and the report command, as shown in Table 3.3. If a packet issued by an IDS passes through with a specific source IP and destination IP, MLID will mix the corresponding watermark with the packet. A control message is composed of three fields that are the message type (i.e., the control code), the target and the argument. A message with 01 as its control code request the LID specified in the target field to starts mixing the watermark with the specific packet flowing through.

When a report command arrives with a certain watermark, say  $\omega$ , the LID returns all of its records concerning  $\omega$  no matter  $\omega$  is found in a out-bounded or an in-bounded packet back to HFIDS. So we can now identify the real intrusion paths and the war zone.

Ctrl code	Target	Arguments
01 (Trace)	Nearest LID	Source IP / destination IP
02 (Report)	All LIDs	Watermark (LIDid + Serial)

Table 3.3 LID control message

Processing a packet involves several components, such as the Network Interface Card (NIC) Driver and the OS, which are not strictly part of the capture architecture. The cost associated with the intervention of such components, in terms of the time they require to process a packet, is called external processing cost. Fortunately, there are some techniques can help to reduce the overheads. LID implement in software can speedup by using some technique, such as Just In Time (JIT) engine and Memory Copies. Besides, LID implement in hardware can use the pipeline technique to improve the performance. With the architecture that Fig.3.6, an LID can execute at most three parallel instructions per cycle.

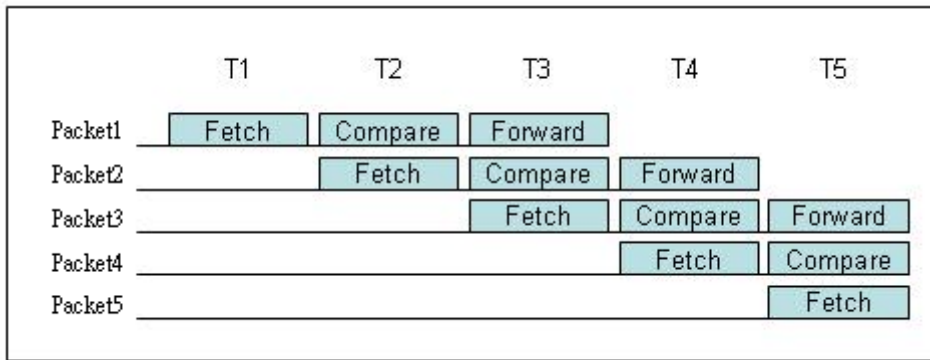


Fig.3.6 A recommended pipeline machine to improve system performance of an LID

### 3.3 Intrusion Scenario

Often, workstations in intranet play an important role in an intrusion or attack as the stepping stones or the inner intruders. Based on the figure 3.7, we give a scenario that may exist and happen in most enterprises and universities that provide servers and hosts, such as Web Server, Mail Server, FTP Server...etc, to service their users.

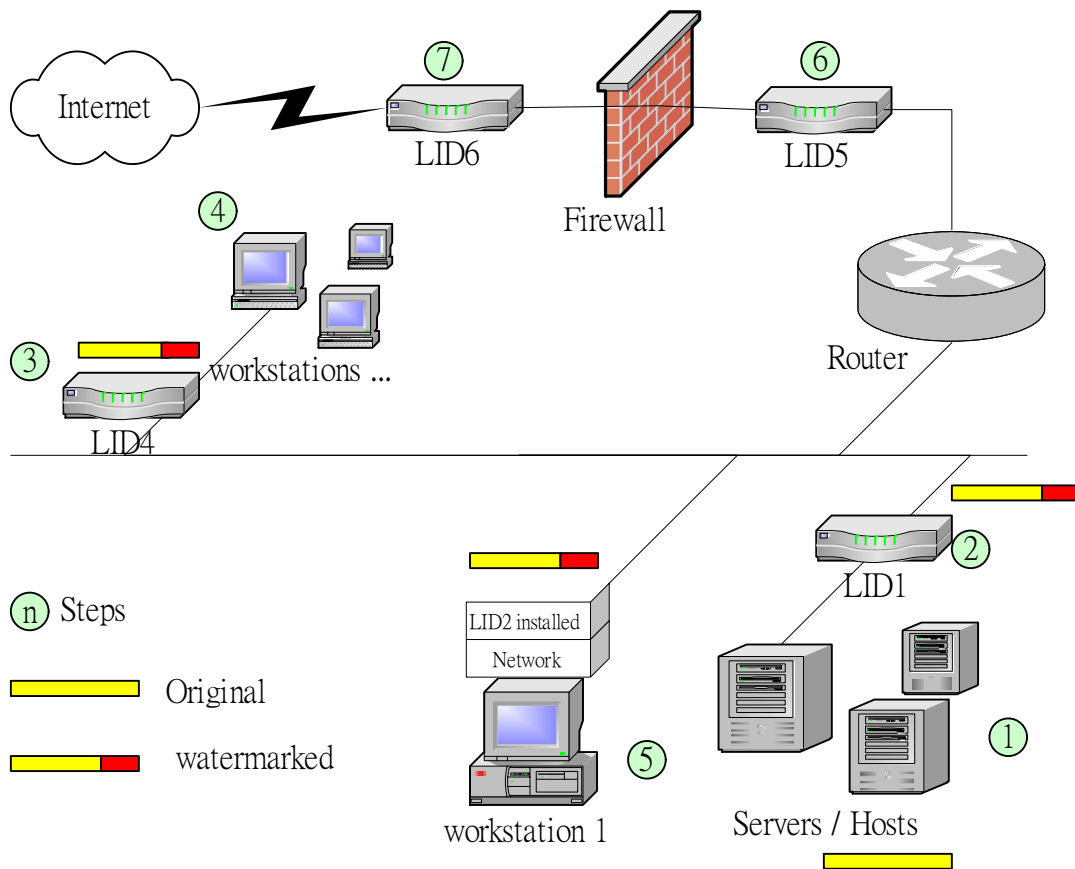


Fig.3.7 Intrusion Scenario

One day afternoon, the IDS installed on the Web server has issued an alarm as being intruded. It decides to request a tracing service for collecting the intrusion paths and then sends a tracing command with LIDP. Now we finish the first step of detecting the intrusion paths. When the nearest LID, say LID1, catch the packet which contain the control message type “01” , then it changes it’s rules to act as a MLID and starts to mix the packets with watermark 0101 from specific source and destination IPs. Now, we finished the second step. Following, the watermarked packets start to travel around our Intranet. When one LID find that a packet with watermark, say 0101, which means that the packets with serial number 01 sent by LID1 pass through, the LID records the watermark and routing paths. Of course, the source IP of the packet is our web server and the destination IP is one of our workstations in another net segment. Then the packet continues its travel to some host/device, say workstation 1. It means that the intruder tries to hide his location by the stepping stones. After tracing a period of time, the IDS decides to reconstruct the intrusion paths in order to terminate the intrusion. It sends a LIDP packet of type 2 to trigger MLID to collect the records LID1 broadcasts a command 020101 to all the LIDs to request them returning their records.

After the sequence of works, by locating the records collected on the topology that is previously established we can find the intrusion paths that start from workstation 1, going through some workstations as the stepping stones and then attempting to intrude the web server.

### **3.4 Algorithms**

Our early research used rule-based and pattern-based databases to identify hackers. The concepts for identifying hackers are not hard to understand. The intrusion detections were sensitive. But as mentioned above, data grew quickly and databases became not maintainable. Query cost was higher time after time. In this article, we develop five algorithms, categorization, sequence analysis, characteristic, mining and comparing to improve the performance of our system by invoking some numerical analysis.



### 3.4.1 Categorization algorithm

The algorithm “Categorization” realizes the real-time host based system since we collect commands from all the users of the system and give each command  $C$  a weight  $f(C)$ ,

$$f(C) = \frac{\text{the times command } C \text{ appear used by all the users}}{\sum_{C_i \in N} \text{the times command } C_i \text{ appear used by all the users}} \quad (1)$$

where  $N$  is the set of commands of an operating system or an application system. Now each command  $C$  has its own representative weight with which all the commands concerned can be classified into two classes, frequently and rarely used. One belonging to the latter may be dangerous.

We sort commands of the system addressed on  $f(C)$  and assign each command a score, say  $S(C)$ , which is an integer. The smaller the  $f(C)$  is, the higher the  $S(C)$  will be.  $S(C)$  is an integer ranging from 1 to 50 representing that  $C$  is a normal command. Duplicated may occur within those commands with lower  $S(C)$ . Once  $S(C)$  is lower than a threshold (e.g., 50),  $C$  becomes less important since it is used almost by all the users. Finally, a fixed high score, e.g., 101, is assigned to all dangerous commands, e.g., `su`, to replace their original scores. Table 3.4 lists some examples of command scores.

Table 3.4 Examples for score table on IM

Command	Score
ELM	3
popd	30
nslookup	1
mount	14
passwd	14
copy	5
su	40 → 101

### 3.4.2 Sequence algorithm

This algorithm is developed to find the sequential-command set (SCS) frequently submitted by a user. A SCS consists of two to five contiguous commands in time sequence. For example, a user U has keyed in three commands “ls → cpine → exit”, i.e., an SCS. Then we look for U’s history file to see if the SCS is a highly repeated pattern or not. If yes, the SCS could be a signature S of U. Furthermore, in U’s history file if there exists a pattern P containing S and S in P do follow S’s original time sequence, e.g., ls → telnet → cpine → ftp → exit, P is also regarded as an S. Please refer to Fig.3.8. In fact, U’s CP is composed of the SCSs submitted by U.

```
Algorithm identifying-SCS

Threshold t1, t2; //a number of consecutive commands, t1 is the lower bound,
and t2 the upper bound

Threshold t3; // a counter that counts the quantity of commands located
between two consecutive commands in the underlying SCS

Recursively Combine (current command sets)

Nested for loop i := 0 to t3 // nested from t1 to t2

Finding commands set(CS) with i separated characters

Combination (CS) //recursively finding the commands belonging to the same
set in the user’s history files to calculate scores
```

Fig.3.8 Sequential command sets algorithm

### 3.4.3 Characterization algorithm

This algorithm assigns a value to each SCS identified as its signature score. When the SCSs of a user have been uncovered, some of them may also exist in other users’ CPs. This may reduce the recognition accuracy. We solve this problem by involving a formula, frequently used in information retrieval for assigning a term a weight [30], to calculate the weight for each SCS. Suppose there is a set of CPs, say  $D = \{CP_1, CP_2, \dots, CP_N\}$ , where N is the number of the CPs of the system addressed,

and  $T = \{S_1, S_2, \dots, S_{M_i}\}$  is the set of SCSs derived from  $D$ , where  $M_i$  indicates the total number of  $S_i$  in  $D$ , i.e.,  $|T| = M_i$ . The weight  $W_{i,j}$  of  $S_i$ ,  $S_i \in T$ , in  $CP_j$ ,  $CSP_j \in D$ , is

$$W_{i,j} = \frac{sf_{i,j}}{sf_{i,j} + 0.5 + 1.5 \frac{ns_j}{AVG(ns_j)}} \times \frac{\log\left(\frac{N+0.5}{M_i}\right)}{\log(N+1)} \quad (2)$$

Where  $sf_{i,j}$  is the frequency of  $S_i$  appearing in  $CSP_j$  and  $ns_j$  the total number of SCSs in  $CSP_j$ .  $AVG(ns_j)$  is the average number of SCSs for each CSP in  $D$  and  $\log(N+0.5/M_i)/\log(N+1)$  the ICPF (inverse characteristics profile frequency). A signature score  $S_{ij}$ , the score of  $S_i$  in  $CSP_j$ , can be obtained by the following formula.

$$S_{ij} = W_{i,j} * 10000 \quad (3)$$

By calculating scores, we can find the relatively important SCSs, and then keep the significant parts and remove the useless based on the threshold given. For each user, the most important SCSs with their scores are stored in his/her own CP (recall, Characteristics Profile) as his/her signatures with which the user can be identified precisely. In our early research, it is hard to decide how many signatures should be kept and how many duplicated portions should be removed. That means, tuning the system to fit different usages and different environments is hard. Now, tuning work becomes more easily and efficiently.

#### 3.4.4 Mining algorithm

The mining algorithm is to compare the users' inputs with their profiles. We need CSP and CP to calculate their similarity with the following formula.

$$SIM = B \times p_1 + \frac{\text{user inputs}}{CP} \times p_2 \quad (4)$$

where  $B$  is a binary variable. When the user's input exists in his/her own CSP, then  $B$

is 1, otherwise, B is 0.  $p_1$  and  $p_2$  are the regulatory probabilities. The higher an SIM is, the more similar the SCS and the CP concerned will be. Now a threshold is needed to determine whether the user is suspected as an intruder or not. All the data, including CSPs, CPs and history files are installed into a data warehouse as the data sources. The history files can be collected from an Operating System, e.g., Sun Solaris System, which record the commands that send by users. The arguments submitted for a query are the sessions and commands collected on line and the command sets stored in CSP and CP. Then by invoking the OLAP, we can find the cooperative intrusions. From the SCS enters and CP, we can not only find if the account P logs in have been misused or not, but also realize who the true intruder is by comparing the SCS with all the users' CPs. Of course, all the CSPs and CPs should be collected and analyzed beforehand.

### 3.4.5 Comparing algorithm

In these chapter, we discuss some method invoke to compare the user behavior and explain how we can find out the different between hackers and users using graph. First we propose a new method Delayed Sequence (DS) Matching method to overcome the comparison with delayed command sequence.

#### DS Matching method

If an input pattern A is a vector whose  $i$ -th element is  $a_i$  ( $i = 1, 2, \dots, I$ ). And an input pattern B is a vector  $b_j$  whose  $j$ -th element is  $b_j$  ( $j = 1, 2, \dots, J$ ). That is,

$$A = a_1, a_2, \dots, a_i, \dots, a_I$$

$$B = b_1, b_2, \dots, b_j, \dots, b_J \quad (5)$$

The correspondence of characteristic parameters of A and B in A-B plane are shown in Figure 3.9. Let F be the sequence of points  $C_k$  ( $i_i, j_k$ ) on the grid.

$$F = C_1, C_2, \dots, C_k, \dots, C_K \quad (6)$$

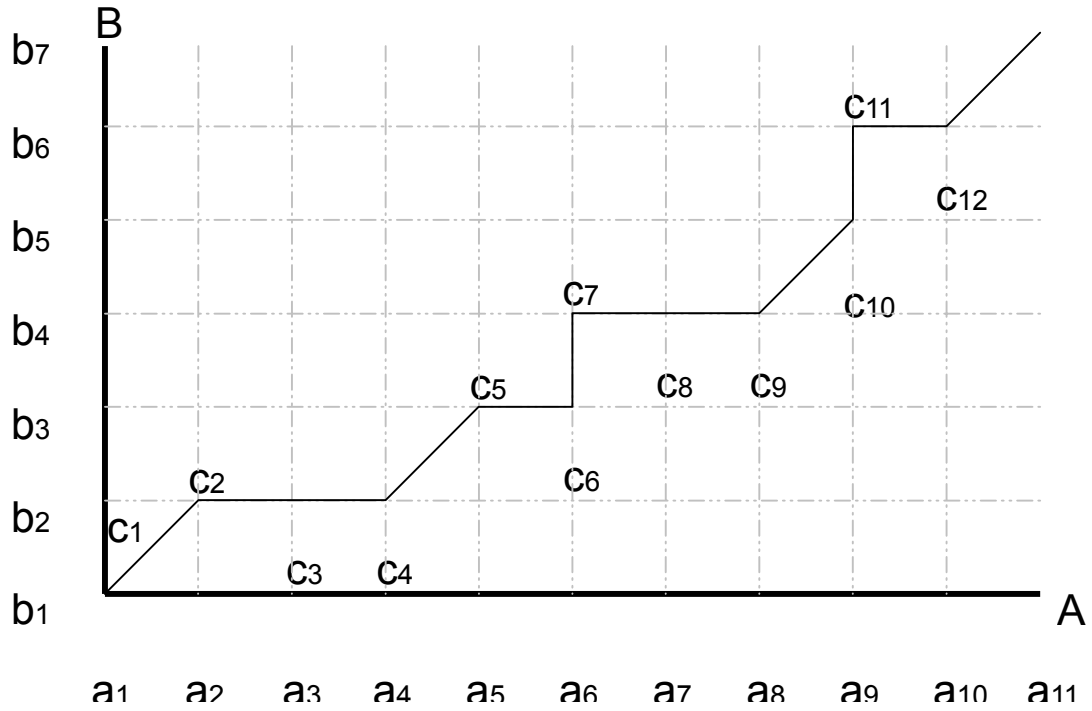


Fig.3.9 Correspondence of characteristic parameters

$a_1$  and  $b_1$ ,  $a_2, a_3, a_4$  and  $b_2$ ,  $a_5, a_6$  and  $b_3$  are corresponding point that represent the matching point. This line is called a path. When a matching of pattern A and B is given by a path  $\omega(i(k); j(k))$ ,  $k = 1, 2, \dots$ , the distance of two elements  $a_{i(k)}$  and  $b_{j(k)}$  is represented as  $d(a_{i(k)}, b_{j(k)})$ . The distance of two patterns  $D(A, B; \omega)$  will be calculated Formula 7:

$$D(A, B; \omega) = \sqrt{\sum_{k=1}^n d(a_{i(k)}, b_{j(k)})} \quad (7)$$

Finding the best matching means finding a path  $\omega$  which minimizes the value of  $D(A, B; \omega)$ . In other words, Formula 8 gives the smallest distance  $D(A, B)$  when we take the best matching.

$$D(A, B) = \min[D(A, B; \omega)] \quad (8)$$

If we define  $g(I, j)$  as shown in Formula 9 and 10 and calculate  $g(i, j)$  iteratively, the distance between two patterns  $D(A, B)$  can be obtained from  $g(I, J)$ , where  $I$  and  $J$  are the length of pattern A and pattern B, respectively.

$$g(i, j) = \min[g(I, j; \omega)] \quad (9)$$

$g(i, j; \omega)$  is the accumulation of paths from (1, 1) to (i, j) when  $\omega((i(1), j(1)), \dots, (i(k), j(k)))$  is given. If  $g(i-1, j)$ ,  $g(i-1, j-1)$ ,  $g(I, j-1)$  are the minimum value of paths for the grids (i-1, j), (i-1, j-1), (i, j-1), the minimum value  $g(i, j)$  of the path to (i, j) can be found as follows Formula 10.

$$g(i, j) = \min \begin{bmatrix} g(i-1, j) + d(i, j) \\ g(i-1, j-1) + 2d(i, j) \\ g(i, j-1) + d(i, j) \end{bmatrix} \quad (10)$$

Figure 3.10 illustrates Formula 10.

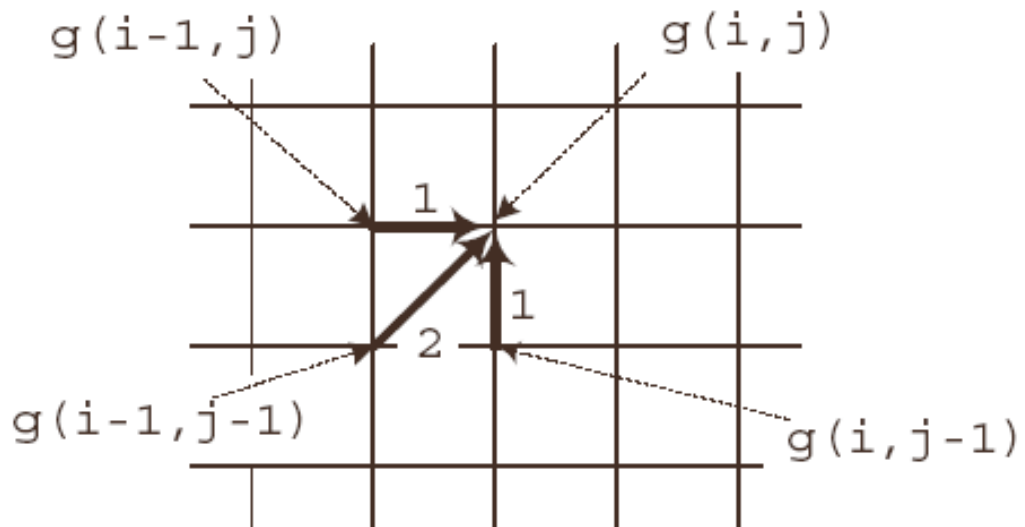


Fig.3.10 Minimum accumulated distance

By adding initial value of  $g(1, 1) = d(a_1, b_1)$ , formula 10 can be calculated recursively. We can find the optimal matching path  $g(I, J)$  of two patterns.

## 4 The experiments

In order to evaluate if PMS is feasible, we collect some history files to do the experiments which have three steps, profile generation, user recognition and an advanced test.

### 4.1 Profile generation

We gather data from a Sun Solaris System that provides mail, web, telnet and ftp services to the graduate students of Tunghai University. There are a total of 4785 students that have frequently used the system. Each student has his/her own home directory and disk quota. When a student U logs in for receiving mails or doing some other work, the system records the commands that U has so far entered one by one in a so-called “history” file, in which a record for the command C has two fields. Please refer to Table 4.1. The first keeps the time that C is entered with a long integer format, whereas the second is the command itself. In the following experiments, we pre-process the data in order to eliminate the useless of the original data so that the performance can be improved. The pre-process has two phases. The first phase gets rid of the first line of each record, i.e., the time, and segments a “session” by identifying login command and its corresponding exit command. Often several commands can be found in a “session”. The second phase discards the segments with less than ten commands and samples the data from the remaining histories. After that, a CSP is created for each user.

Table 4.1 An example of history file on Solaris

```
#+1058386787
exit
#+1058713621
ver
#+1058713626
man ver
#+1058713636
....
```

## 4.2 User Recognition

The procedure of recognizing a user from his/her inputs and profiles has four steps.

**Step 1:** Each command  $C$  is given a score,  $S(C)$ , by invoking the algorithm “Categorization”. A score table is then created and the scores of the dangerous commands, as mentioned above, are replaced with some fixed high scores, which are often higher than 100. Please refer to Table 3.1.

**Step 2:** This step explores the feature of a command set and creates a CSP for each user based on the score table constructed in step 1 and the algorithm “Sequence analysis”. About 80% of the user’s history data are used to create user profiles. The remaining 20% will be the test data in step 3 for an accurate evaluation of the recognition accuracy.

**Step 3:** We assign each SCS in the CSP a score also with the algorithm “Characterization”. SCSs now become signatures. Table 4.2 gives an example of a CP in which each line is a signature. The commands are the corresponding SCS. The rear value is the SCS relatively important score. Now, the recognition efficiency can be dramatically improved due to the quantified value in stead of string comparisons.

Table 4.2 An example of CP

telnet -> cpine -> exit = 598
telnet -> cpine -> telnet -> telnet -> exit = 5187
cpine -> quot -> exit = 660
cpine -> vi -> cpine -> vi -> exit = 2660
.....
dropped
cpine -> exit = 56
telnet -> exit = 89

Table 4.2.1 shows that “cpine” (pine command of traditional Chinese version) appears in all of the lines indicating that the user prefers invoking mail services. That is, this user frequently logs in the mail server, receives mails, then logouts. Besides, telnet, quot, vi and exit are the commands he/ she has ever used.



**Step 4:** This step is to evaluate the accuracy of recognition. We use the mining algorithm and DNA comparison algorithm to calculate the similarity. We use the 20% of the user’s history data to simulate the user inputs and compare them with all the user profiles and finally sort the result with the similarity values. There are 292,010 commands in our 4785 samples. The recognition rate is 75.64% in average. The specifications of the hardware platform are listed in Table 4.4.

Table 4.3 The time required to finish the experiment (A total of 292,010 commands from 4785 users).

Programs	Time (min)	Efficiency (Sec/User)
Pre-Processing	0.165	0.002
step 1	0.188	-
step 2	12.166	0.342
step 3	21.286	0.683
step 4	42.217	1.187

Now, please refer to Table 4.3 The pre-processing gets rid of the unsuitable data (e.g., commands that less then 10). Step1 generates the score table. Step2 generating a CSP and the test file for each user only takes 12 minutes for 2134 users, i.e., 0.342 second /per user. Step3 characterizing each user’s CP takes only 21 minutes for 2134 users, i.e., 0.683 second/person. Step 4 uses the users’ test file to test against the CP files for recognition. 42 minutes are needed for 2134 users, i.e., 1.187 second/per user. Hackers of cooperative attacks are also explored in this step.

Table 4.4 Specifications of the hardware platform

component	description
CPU	AMD ATHON XP 1.7G
RAM	512MB DDR400
HDD	Seagate barracuda IV 80G
OS	Windows .net server 2003
Software	j2sdk1.4.2

### 4.3 Recognition trend with number of commands

We choose different number of commands in history file for each user and find out that the trend of the recognition rate against the change of commands. Please refer to Figure 4.1 Without invoking pre-processing, the test data come from 4785 users. The recognition rate is 75.64%. After pre-processing is performed, 2134 users remain, each with at least 10 commands. The recognition rate is 78.84%. We remove those users with less than 50 commands. Now 1177 users satisfy the limitation and the recognition rate is 91.82%. When raising to at least 100 commands. Only 751 users fulfill the constraint and the recognition rate is 93.18%. Finally, 307 users each has at least 200 commands generate a 95% recognition rate. Please refer to Table 4.5 and Fig 4.1

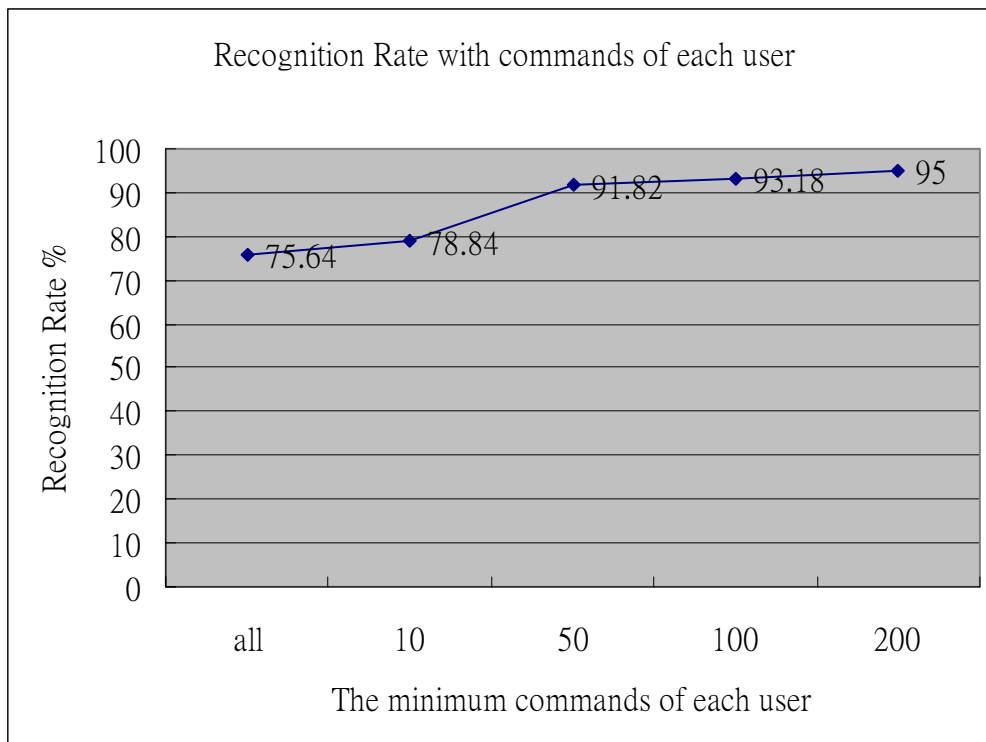


Fig.4.1 The trend of recognition rate with the minimum commands for each user

Table 4.5 The detail of recognition rate with the minimum commands for each user

	users	Recognition rate
All	4785	75.64%
10	2134	78.84%
50	1177	91.82%
100	751	93.18%
200	307	95%

#### 4.4 Recognition trend with number of users

Fig.4.2 shows the trend of recognition rate with the growing of users. We can see that the trend decrease slowly. Comparing with that in Fig.4.1, the recognition rate of our system is more sensitive with the commands per user than with the number of users.

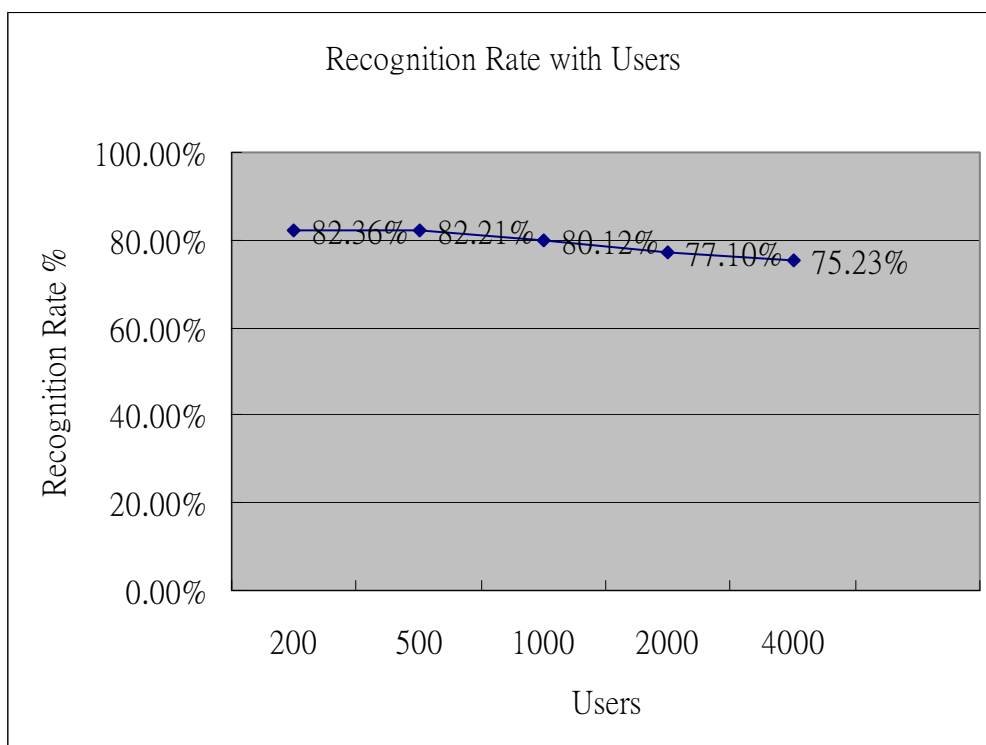


Fig.4.2 The trend of recognition rate with the growing of users

#### 4.5 Recognition rate with specific clusters

We choose the undergraduate students that study in computer science and information engineering (CSIE) for test as the second experiment. There are 440 students with 95631 commands. The average commands per user (ACPU) is 217. The recognition rate is 84.84%. Please refer to Table 4.6,

Table 4.6 Recognition rate with CSIE students

Item	Value
users	440
commands	95631
ACPU	217
Recognition rate	84.84%

The third experiment is performed upon the students major in mathematics. There are a total of 160 students with 9096 commands. ACPU is 56.625. The recognition rate is 73.55%. Please refer to Table 4.7,

Table 4.7 Recognition rate with Math students

Item	Value
Users	160
Commands	9096
ACPU	56.625
Recognition rate	73.55%

The fourth experiment is performed upon the students studying in Industrial Engineering Department. There are 139 students with 7257 commands. ACPU is 52.21. The recognition rate is 74.89%. Please refer to Table 4.8,

Table 4.8 Recognition rate with Industrial Engineering Department

item	Value
users	139
commands	7257
ACPU	52.21
Recognition rate	74.89%

The results show that the recognition rates are much more sensitive with the ACPU than with the number of users or with some specific domain.

#### 4.6 An advanced Test

We choose two users, A and B, who use almost the same commands/programs to do their works (e.g., telnet, elm, pine...), i.e., their CSPs are almost the same, for an advanced test. A's history file is invoked as the inputs to test B's CP and CSP and vice versa. Let  $p_1 = 30$  and  $p_2 = 70$  for formula (4). The recognition rates of user A is 90.91% and User B is 98.03%. It means that each has his/her own habits which are quite different from the other one's. The similarity calculated by formula (4) is 35.54%. It means that CP works and can uniquely identify one person with the other similar one. Fig. 4.3 shows the comparison with A and B in human visualization.

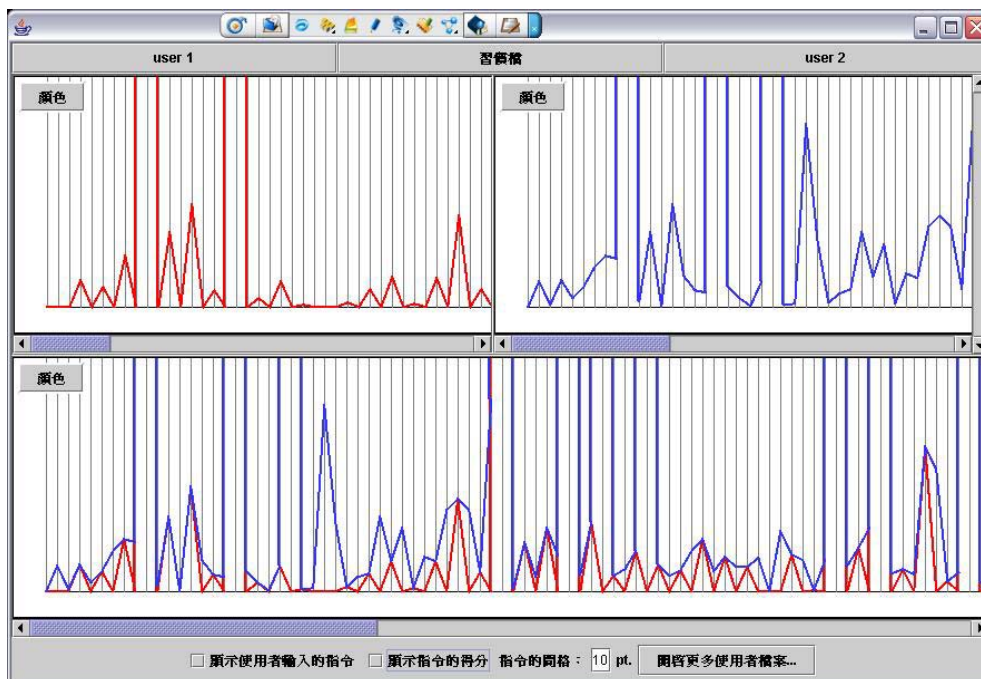


Fig. 4.5 Comparison with user A and B in Graph

## 5 Conclusion

In this research, we propose an approach to collect users usage histories and evaluate each command for giving it a score. Then catch the features of a user from the commands he/she was used to submit. The main idea is about transforming the text into numbers, then we can deal with those data efficiently. In order to prove our ideas and algorithms, several experiments has been done. First, we left the data originally without any processing and the recognition rate is only 75%. One of the main reasons is that there are lot of the user may only login once. The others maybe due to the wrong commands and the poor kinds of commands that have ever been send. By doing some pre-process to the data, dropping those users who use less then fifty commands. The recognition rate raise to 90%. The result is expectable and acceptable. We have successfully found out the differences between users. Besides, we did future tests by choosing three different domain users to evaluate whether the algorithm is sensitive to the kinds of commands or the number commands that have been send. The result show that it is more depend on the ACPU rather then the number of users or the kinds of commands been send. Another test is done in order to prove the algorithm can find out the habit of the user. We choose two similar users, who have almost the same CSP. It means that they use the same command to finish their jobs. And the result is that we can still find out their difference only by their using habits. The performance is also acceptable.

In the advanced researches, we will try to use different technologies to improve the recognizing rates, such as regressive function, DNA algorithm, neural networks and artificial intelligent. Regressive function can help us to find out the intrusion models. DNA algorithm can helps us to do the sequence analysis because the algorithm can deal with the data that have no directions. Neural networks can help PMS to recognition the user in another way. Finally, the AI can help RFIDS to decide more precisely and decrease the false alarm rates. But the efficiency and latency delay may be the problems to be solved.

## Reference

- [1] 2003 CSI/FBI Computer Crime and Security Survey. <http://www.gocsi.com/>
- [2] J. Saltzer and M. Schroeder, "The Protection of Information in Computer Systems," Proc. IEEE, vol. 63, no. 9, 1975, pp.1278-1308.
- [3] Anderson, James P. Computer Security Technology Planning Study 2. ESD-TR-73-51, Bedford, MA: Electronic Systems Division, Air Force Systems Command, Hanscom Field, October 1972.
- [4] T. F. Lunt et al., "IDES: A Progress Report," Proc., Sixth Annual Computer Security Application Conf., Tuscon, AZ, Dec, 1986.
- [5] T. F. Lunt et al., "A Real-time Intusion Detection Expert System (IDES)," Interim Progress Report, Project 6784, SRI International. May 1990.
- [6] H. S. Javitz and A. Valdez, "The SRI IDDES Statistical Anomaly Detector," Proc., 1991 IEEE Symposium on Reserach in Security and Privacy, Oakland, CA, May 1991.
- [7] D. E. Denning, "An intrusion detection model," IEEE Trans. On Software Engg., vol. SE-13, pp. 222-232, Feb. 1987.
- [8] Halme, Lawrence, T. Lunt, and J. Van Horne. "Automated Analysis of Computer System Audit Trails for Security Purposes. "Proceedings of the National Computer Security Conference, Washington, D.C., September 1986.
- [9] Tener, William T. "Discovery: An Expert Ststem in the commercial Data Security Environment." Proceedings of the IFIP Security Conference, Monte Carlo, 1986.
- [10] Smaha, Stephen E. "An Intrusion Detection System for the Air Force." Proceedings of the Fourth Aerospace Computer Security Applications Conference, Orlando, FL, December, 1988.
- [11] Sebring, Michael M., E. Shellhouse, M. E. Hanna, and R. A. Whitehurst. "Expert Systems in Intrusion Detection: A Case Study. "Proceedings of the Eleventh National Computer Securitiy Conference, Washington, D.C., October 1988.
- [12] Hochberg, J. et al. "NADIR, An Automated System for Detecting Network Intrusion and Misuse." Computer and Security 12, no. 3, May, 1993, 235-248.
- [13] Heberlein, L. T. "A Netowrk Security Monitor." Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy, Oakland, CA, May 1990, 296-304.
- [14] Vaccaro, H. S. And G. E. Liepins. "Detection of Anomalous Computer Session

- Activity. “Proceedings of the 1989 IEEE Symposium on Research in Security and Privacy, Oakland, CA, May, 1989, 280-289.
- [15] Snapp, S. R. et al. “DIDS (Distributed Intrusion Detection System) – Motivation, Architecture, and An Early Prototype. “Proceedings of the Fifteenth National Computer Security Conference, Baltimore, MD, October, 1992.
- [16] A. K. Ghosh, J. Wanken, and F. Charron, “Detecting Anomalous and Unknown Intrusions Against Programs,” Proc. Annual Computer Security Application Conference (ACSAC’98), IEEE CS Press, Los Alamitos, Calif., 1998, p.259-267.
- [17] K. Ilgun, R. A. Kemmerer, and P. A. Porras, “State Transition Analysis: A Rule-Based Intrusion Detection System,” IEEE Trans. Software Eng. Vol. 21, no.3, Mar, 1995, p.181-199.
- [18] U. Lindqvist and P. A. Porras, “Detecting Computer and Network Misuse with the Production-Based Expert System Toolset,” IEEE Symp. Security and Privacy, IEEE CS Press, Los Alamitos, Calif., 1999, p.146-161.
- [19] <http://www.microsoft.com/WindowsServer2003/iis/default.msp>
- [20] F.Y. Leu, T.Y. Yang. “A Study of IIS Web Server Vulnerabilities,” Proceedings of the 5<sup>th</sup> Information Management and Police Administrations Conference, TauYuan Taiwan, June 2001, pp. 94-100.
- [21] Ye, N.; Zhang, Y.; Borrer, C.M., “Robustness of the Markov-Chain Model for Cyber-Attack Detection.” IEEE Transactions on , Volume: 53 , Issue: 1 , March 2004, Pages:116 - 123
- [22] Kreidl, O.P.; Frazier, T.M., “Feedback Control Applied to Survivability: A Host-Based Autonomic Defense System.” IEEE Transactions on , Volume: 53 , Issue: 1 , March 2004, Pages:148 - 166
- [23] J.E. Dickerson and J.A. Dickerson, “Fuzzy Network Profiling for Intrusion Detection,” Proc. of Fuzzy Information Processing Society, 2000.
- [24] W. Lee, S.J. Stolfo, and K.W. Mok, “A data mining framework for building intrusion detection models, “IEEE Security and Privacy, 1999.
- [25] Lawton, G., Computer , “Biometrics A new era in security,” Volume: 31 , Issue: 8 , Aug. 1998, Pages:16 – 18.
- [26] Y. Okazaki, I. Sato and S. Goto, “A new intrusion detection method based on process profiling,” Applications and the Internet, 2002. (SAINT 2002). Proceedings. 2002 Symposium on Jan, 2002, Pages: 82-90.
- [27] F.Y. Leu, T.Y. Yang. “A host-based real-time intrusion detection system with data



mining and forensic techniques,” Proceedings of the IEEE 37th Annual 2003 International Carnahan Conference, Oct, pp. 580-586.

[28] <http://www.netscape.com/>

[29] X. Wang, D. Reeves, S. F. Wu, and J. Yuill, "Sleepy Watermark Tracing: An Active Network-Based Intrusion Response Framework", Proceedings of IFIP Conference. on Security, Mar. 2001.

[30] A. Leuski, "Evaluating document clustering off interactive information retrieval," ACM CIKM'01, November 2001, pp. 33-40.