國立交通大學

資訊科學系

碩 士 論 文

一個為阻絕服務攻擊的蠕蟲免疫服務專家系統

A Worm Immune Service Expert system for denial of service attacks

研 究 生：吳政霖

指導教授：曾憲雄　教授

中 華 民 國 九 十 四 年 六 月

一個為阻絕服務攻擊的蠕蟲免疫服務專家系統

研究生: 吳政霖　　　　　　　　　　　　　　指導教授: 曾憲雄博士

國立交通大學資訊科學系

摘要

隨著網路快速的發展，蠕蟲感染和傳播的速度也隨之增快．除此之外，新型變種蠕蟲的產生也越來越快．因此在本論文中我們利用 VODKA 來幫助專家容易的找出這些變種蠕蟲．VODKA 是一個發現變異物件的知識擷取方法，找出隱藏在實際世界中的變異物件。另外，目前網路關於蠕蟲的技術文件大都缺乏結構化，所以想要透過資料探勘的方式找出知識是不容易的，因此，我們藉由兩階段知識擷取，先建構概念階層，再透過階層式表格法進行蠕蟲的知識擷取動作，最後建構出整體系統的蠕蟲知識庫。

通常當使用者的系統發生異常狀況時，使用者常常會藉由察覺到的徵狀到搜尋引擎找尋相關資料也因此而浪費了不少時間．為了幫助使用者能夠夠容易的發現蠕蟲危害，我們提出一個為阻絕服務攻擊的蠕蟲免疫服務專家系統並且也完成包含三個模組的系統之實作．此三個模組包含了：診斷模組、教學模組和學習模組，分別用來幫助使用者偵測系統的安全狀況、教導使用者在面對蠕蟲時如何防禦和透過 VODKA 來協助專家找出變種蠕蟲．

關鍵字：蠕蟲，知識擷取，變異物件，知識庫，專家系統．

WISE: a Worm Immune Service Expert system for denial of service attacks

Student: Cheng-Lin, Wu Advisor: Dr.Shian-Shyong Tseng

Department of Computer and Information Science
National Chiao Tung University
Hsinchu, Taiwan, 300, Republic of China

Abstract

With the rapid development of Internet, the worm can spread and infect other computers quickly. Besides, new variant worm is evolving too fast, so we need some efficient approaches to discover it. Therefore, we use VODKA approach to help experts discover these variant worms easily in this thesis. The Variant Object Discovering Knowledge Acquisition (VODKA) is a method of finding a new object from the inference results. And most of worm technological documents are non-structured, so discovery of knowledge by data mining is not easy. We use a Two-Phase Knowledge Acquisition methodology is proposed to acquire the concept hierarchy of worm first and then to extract knowledge of worms based upon hierarchical repertory grids adjustment from experts. Last, we will construct the knowledge base of worm.

When worm infects the system, the users perceive some abnormal behaviors. The users usually will usually input what he or she looks to the search engine on the network and looks for the solution. Such a way consume large time look for correct solutions. Consequently, we propose and implement a worm immune service expert system (WISE) to the user. WISE contains three modules: diagnosis module, tutorial module and discovery module for detecting the security status of machines, teaching users how to defense threat of worms and learning variant worm by VODKA respectively.

Keywords: worm, knowledge acquisition, VODKA, knowledge base, expert system.

致謝

能順利完成本篇論文，最重要的必須感謝我的指導教授，曾憲雄博士，曾教授在我碩士班兩年期間相當耐心的指導我論文研究;從他身上我也學習許多領導處事的技巧，這些寶貴的經驗讓我獲益匪淺，感激不盡。同時也感謝我的口試委員，黃國禎教授、陳年興教授和蔡文能教授，他們給予了我相當多的寶貴意見，讓本篇論文更有價值。

第二位要感謝的人是林順傑學長，兩年期間讓我學會許多理論知識及實務技巧，也給予了我許多對於此篇論文的寶貴意見，接受我的詢問和討論，和協助我論文修改工作，深表感激。

此外我必須感謝實驗室學長平日的諸多幫助，同時也感謝實驗室同窗夥伴，黃柏智、李育松、陳君翰、宋昱彰、陳瑞言、邱成樑、林易虹等人在生活上和課業上互相幫忙的情誼；以及學弟鄧嘉文等人在論文實驗上的支援，深表感激。

最後我要感謝我的家人對於我的支持與鼓勵，讓我在面對挫折的時候能夠繼續向前，也讓我能夠有相當的自信完整本篇論文，深表感激。

Table of Content

List of Figures

List of Tables

List of Examples

List of Algorithms

Chapter 1
Introduction

An Internet worm is a program that self-propagates across network exploiting security or policy flaws in widely used services. It is not a new phenomenon, having first gained widespread notice in 1988 [12]. We distinguish between worms and viruses in that the latter infect non-mobile files and therefore require some sort of user action to abet their propagation. As such, viruses tend to propagate more slowly. They also have more mature defenses due to the presence of a large anti-virus industry that actively seeks to identify and control their spread. As we know, Internet worms and denial of service (DoS) attacks have had a significant impact on businesses in recent years. Most of enterprises adopt anti-virus software and hardware to protect important information. But, when an unexpected worm appears, those protective mechanisms can't guarantee to protect perfectly. Then the new worm would infect the enterprise and serious losses may result.

With the rapid development of Internet, the worm can spread and infect other computers quickly. Besides, new variant worm is evolving too fast, so we need some efficient approaches to discover it.The purpose of this thesis is to present a Worm Immune Service Expert system (WISE) to diagnose host health and recovery from infected as soon as possible. In addition, the contributions of this thesis are listed as follows:

1) Construct worm knowledge base by two-phase knowledge acquisition

There are many technique documents about worm from Internet. Those documents do not have formal structure, two-phase knowledge acquisition methodology is hence applied to construct worm knowledge base.

2) Design and implement the WISE system.

This thesis describes the theoretical foundations of a system that discovers variant objects using VODKA [15] and a practical approach of its implementation. Finally, we build an environment of web service including the abilities of diagnosis, tutorial, and learning to interact with users by collecting the significant worm information to detect and protect the system security.

The rest of thesis is organized as follows. Chapter2 surveys the life cycle of worm and the background knowledge of this work. Chapter 3 describes the whole architecture and introduces three parts of the Worm Immune Service Expert system (WISE) for denial-of-service attacks. Chapter 4 to Chapter 5, the details of worm knowledge base construction and WISE implementation, are described, respectively. Chapter 6 gives conclusions of this work.

Chapter 2
Related Work

The infection speed of the worm and the appearance frequency of the new worms are quickly and shortly since the network becomes faster and faster. It becomes one of the important issues to efficiently detect and defend the intrusion of the worm. Before proposing our methods to defend those malicious worms, the life cycle of computer worm will be presented in Section 2.1 and the background of worm will be then introduced in Section 2.2. Section 2.3 compares the detection methods of the worm that consist of host-based and network-based detection. Besides, some kinds of worm defense products are provided on the market now, but which one is suitable to be used for defending the worm. Therefore, all the advantages and disadvantages of general antivirus softwares will be introduced in last Section.

2.1: Worm Life Cycle

As mentioned above, in order to understand the threat posed by computer worms, it is necessary to understand the life cycle of worm [5] shown in Figure 2.1. Each worm agent begins with an Initialization Phase. This phase includes things like installing software, determining the configuration of the local machine, instantiating global variables, and beginning the main worm process. For a worm to infect a machine, it must first discover that the other potential victim exists in Target Selection Phase. There are a number of techniques by which a worm can discover new machines to exploit like port scanning. Then a worm can either actively spread it from machine to machine, or it can be carried along as part of normal communication through the network in Network Reconnaissance Phase. After that, infected system may appear many abnormal behaviors to attack user host in Attack Phase and infect next victims in Infect Phase. Besides, different sorts of attackers will desire different payloads to directly further their goals, so that make detect more hard. Therefore, based upon the lifecycle of worm, we can identify the network propagating stage, attacking stage, and infecting stage to help users to improve the security of system.

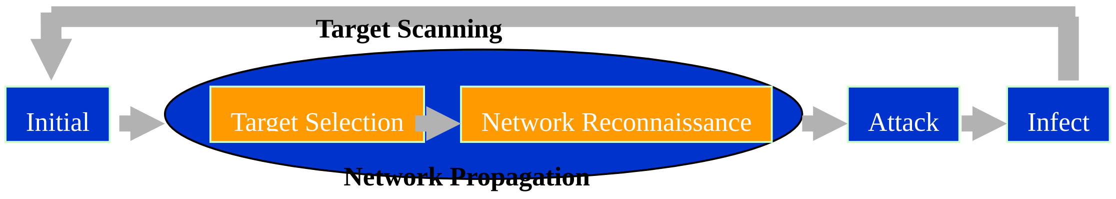


Figure 2.1 Worm Life Cycle

2.2: Brief of Worms

The following list gives an idea of the equalities and differences between some of the prevailing definitions of worm and virus [13].

Virus: A self-replicating program. Some definitions also add the constraint that it has to attach itself to a host program to be able to replicate. Often viruses are said to reside only on the infected host computer, not replicating outside it.

Worm: Also a self-replication program, which does not need another program to be able to replicate but instead is a stand-alone program. And usually performs malicious actions, such as using up the computer's resources and possibly shutting the system down. The difference between a worm and a virus is often said to be the way they replicate, worms replicate over network connections, while viruses replicate on the host computer.

We can know that the worm causes harm of system and Internet is bigger and quicker than virus; therefore this thesis will focus on detecting the worm. Now, several kinds of worm will be introduced and used to built in our knowledge base

In July 2001, the Code Red Worm [2] was released on the Internet. Code Red affected Microsoft Index Server 2.0 and the Windows 2000 Indexing service on computers running IIS 4.0 and 5.0 Web servers. The worm sent its code as an HTTP request on port 80, which exploited known buffer-overflow vulnerability. The worm also attempted a Denial of Service (DoS) against www.whitehouse.gov (198.137.240.91) if the date was between the 20th and 28th of the month.

Code Red used a random number generator to get new victim IP addresses to attack. The initial revision of Code Red hits the same machines over and over again which limited the worm’s ability to spread. Code Red II used a better random number generator to create more target IP addresses by keeping the network portion of the IP address, and then choosing a random host portion of the IP address. This allows the worm to spread itself faster within the same network.

A few months after Code Red was released, the Nimda worm [3] surfaced. It took advantage of some similar vulnerabilities as Code Red, however, it was a hybrid attack that contained both worm and virus characteristics. As a more advanced attack, it could infect more systems and could infect systems in multiple ways. Nimda could infect any computer running Microsoft Windows software by exploiting a flaw in Outlook Express and known vulnerabilities in Microsoft's Internet Information Services software (IIS) 4 or 5, including the security hole left by Code Red II.

Nimda used randomly generated IP addresses to target vulnerable IIS servers using TCP 80 (HTTP). It copied itself to the vulnerable web server as a DLL using TFTP (UDP port 69), then created a listening port ready to transfer the copy of the worm. The transfer of the worm used NetBIOS TCP ports 137-139 or TCP port 445. Nimda then searched for all open network shares using Network Neighborhood. Nimda also used TCP port 25 (SMTP) for sending email to other systems using addresses taken from the infected system.

The Slammer worm [10] was released in January 2003. The primary impact of Slammer was a consumption of network bandwidth. It infected systems at a rate not seen before and saturated networks quickly. The worm created a Denial of Service attack due to the large number of packets it sends. In some cases, networks experienced 100% packet loss.

Slammer targeted systems running Microsoft SQL Server 2000 and Microsoft Desktop Engine (MSDE) 2000. It took advantage of a buffer overflow vulnerability that allowed a part of system memory to be overwritten, and then ran in the same security context as the SQL Server service. Slammer used UDP for connection setup so it did not have the overhead of the connection setup and management required by TCP-based services. Code Red and Nimda took advantage of flaws in TCP-based services, which required a full three-way handshake before exchanging data. Slammer flooded the network by continuously sent traffic consisting of 376 byte packets to UDP port 1434 of randomly generated IP addresses.

In August 2003, the Blaster worm [11] hit the Internet, targeting Microsoft Windows 2000 and XP systems in an attempt to take advantage of a recent published Microsoft Windows RPC DCOM vulnerability. The Distributed Component Object Model (DCOM) allows Microsoft software to communicate. This includes communication with Internet protocols such as HTTP. A flaw in the RPC code was exploited to cause a buffer overflow and the RPC service would fail. This allowed an attacker to run code with System privileges which included the ability to install programs, change data and create accounts with administrator rights.

Blaster used several ports on compromised systems as listening ports or to execute commands. It scanned a random IP range looking for systems to infect on TCP port 135, which is the port that the RPC process listens on. The traffic created by

Blaster saturated subnets with port 135 requests. TCP port 4444 was also a listening port that allowed an attacker to issue remote commands through a hidden shell process. UDP port 69 listened for requests to transmit the virus from computers that were compromised by the RPC DCOM vulnerability. The worm also sent 40 byte HTTP packets on port 80 to windowsupdate.com at the rate of 50 packets per second. If it was unable to find a DNS entry for windowsupdate.com, Blaster used a broadcast address of 255.255.255.255.

As mentioned above, we can know that worms always exploit vulnerable service to attack the victims. And these victims will generate many abnormal behaviors to denial of service in bandwidth consumption and resource consumption.

2.3: Host-based and Network-based Detection

As we know, there are two different types of technique in worm detection: Host-based detection system and Network-based detection system.

Host-based detection systems (HDS) can be self-contained, sending alarm information to the local console, or remotely managed by a manager/collector that receives periodic updates and security data. A host-based implementation that includes a centralized management platform makes it easier to upgrade the software. HDS is ideal if a limited number of critical systems need protection, and they are complementary to network-based detection system.

Network-based detection system (NDS) monitors activity on a specific network segment. Unlike HDS, NDS are usually dedicated platforms with two components: a sensor that passively analyzes network traffic and a management system that displays alarm information from the sensor and allows security personnel to configure the sensors.

The basic premise is that it takes multiple layers of defense to protect against the wide variety of attacks and threats. Not only can one product or technique not protect against every possible threat, therefore requiring different products for different threats, but having multiple lines of defense will hopefully allow one product to catch things that may have slipped past the outer defenses. The following are example of NDS and HDS solution.

Access Control Lists to Protect a Network from Worm/DoS Attacks [6] is one kind of network-based detections. The use of access control lists (ACL's) at the router level is a critical network security practice to safeguard a network infrastructure from worm/DoS attacks. ACL are presented for general monitoring and blocking of malicious traffic, logging of potentially malicious traffic, blocking infected hosts, filtering out malicious traffic from mission critical systems, and an emergency stop ACL to block a significant worm/DoS attack. Though network-based detections can avoid the diffusion of worm, it only relieves worm propagation.

General antivirus softwares that are like PC-cillin of Tread Micro and antivirus of Symantec belong to host-based detection that find infected hosts and remove malicious worm, so it can exterminate the worm propagation. There are network-based detection systems (NDS) and there are host-based detection systems (HDS). While it can be more expensive to implement HDS- especially in a large, enterprise environment, I recommend host-based security wherever possible. Stopping intrusions and infections at the individual workstation level can be much more effective at blocking, or at least containing, threats.

Antivirus softwares that defend not only virus but also worm usually scan worm through pattern matching. In general, antivirus softwares are good at detecting, but they shouldn't be relied upon to provide a total defense. Many variant worm incidents

Therefore there are many weaknesses in general antivirus softwares. First, a great deal of antivirus softwares are inclined to scan the systematic state automatically, users do not know whether it is the correct solving the problem that the system appears or not. Second, general antivirus softwares usually consider the factors in single viewpoints like file scanning or system loophole scanning to warn users of the danger. Third, general antivirus softwares need experts to find variant worm through experimentations that is wasting time.

In order to enhance these weaknesses, this thesis provides some methods. We build a web environment of interacting with users and make the users faster finding worm information, so the users can input symptoms what he found and save the time of search worm relevant information in Internet. And we consider the factors in multiple aspects including file system, system loophole and network flow, etc. Hence, the result of diagnosis can accurately know whether the system is being infected by the worm or not, then we will tell the user how to kill the worm; therefore users can study more knowledge about worm. Besides, we can assist expert to find variants via learning module so it make experts to save a lot of time to experiment.

With the technology revolution, mankind's ability to generate new data has rapidly increased. But our capability to analyze knowledge from this mass of data could not keep up with the knowledge explosion. As we know, a knowledge system is an intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solution. In order to construct our worm knowledge base, we use Embedded Meaning Capturing and Uncertainty Deciding (EMCUD) [9] to interact with experts.

EMCUD can be used to elicit the embedded meanings of knowledge from the existing hierarchical repertory grids. The details are shown in Section 2.4.

2.4: EMCUD

In the past few years, many knowledge acquisition systems were developed to rapidly build prototypes and to improve the quality of the elicited explicit knowledge. The traditional approach of knowledge acquisition from domain experts is via interviewing. However, deeper knowledge can be elicited using knowledge acquisition systems. Up until now, several models have been proposed for handling uncertainties in expert systems. One of the most fruitful models of uncertain reasoning is the EMYCIN Certainty Factor (CF) model. Furthermore, the CF model which was first used in the medical expert system EMYCIN [7] decides the degree of belief of a rule. To extract rules with embedded meaning, Embedded Meaning Capturing and Uncertainty Deciding (EMCUD) [9] knowledge acquisition based on Personal Construct Theory [8] was proposed. *EMCUD* can be used to elicit the embedded meanings of knowledge from the existing hierarchical repertory grids. Additionally, it will also guide experts to decide the certainty degree of each rule with embedded meaning for expending the coverage of generated rules. To capture the embedded meanings of the resulting grids, the Attribute Ordering Table (AOT), which is used to record the relative importance of each attribute to each object, is employed. There are three different values in each AOT entry, a pair of attribute and object, "X", "D" or an integer; "X" means no relationship existing between the attribute and the object; "D" means that the attribute dominates the object; integer is represented for the relative important degree of the attribute to the object. The larger integer number means the attribute is more important to the object.

Using AOT [9], the original rules generate some rules with embedded meaning, and the Certainty Factor (CF) of each rule, which is between -1 and 1, could be determined to indicate the degree of supporting the inference result. The higher CF is, the more reliable the result is. The *EMCUD* algorithm is listed as follows.

Algorithm 2.1: *EMCUD* algorithm

```
Input: The hierarchical grids.
Output: The guiding rules with embedded meaning.
Step1: Build the corresponding AOT with each grid of the hierarchical multiple grids.
Step2: Generate the possible rules with embedded meaning.
Step3: Select the accepted rules with embedded meaning through the interaction with experts.
Step4: Generate automatically the CF of each rule with embedded meaning.
```

To decide the CF of each embedded rule, we have to firstly decide the upper and the lower bounds of CF values of accepted embedded rules. CF values of each rule can be automatically determined by fuzzy mapping function. Thus, the useful embedded rules with corresponding CF values could be used to cover more uncertainty cases.

Chapter 3
The Framework of WISE

As we know, there are many antivirus softwares that can discover worm, virus or Trojan horse in our system. Although these antivirus softwares are developed to protect our system well, it is hard to automatically discover the new worm without updating their signature database. In order to overcome the weakness, the Worm Immune Service Expert system (WISE) is proposed to enhance the commercial antivirus products with their well-performance instead of replacing them. WISE is a knowledge-based system. Unlike pattern matching system, it does not need to write the program again, and therefore is suitable for worm, which is usually variant quickly that updates knowledge base frequently. Besides, WISE contains embedded meanings of knowledge that will be introduced in next chapter, so WISE can easily capture some variant worms that in order to avoid signature-based detection system to modify characteristic less.

In order to diagnose precisely, we will define system statuses that describe in Section 3.1. Section 3.2 will model the knowledge class of worm according to the worm life cycle. Section 3.3 will introduce the architecture of WISE.

3.1: Definition of Host Status

According to the life cycle of worm, we know that worms have distinct behaviors in different stages. In order to realize threat of worm, we model system status into five states that is proposed to diagnosis system security level in real world. The system status diagram is shown in Figure 3.1.

The N and C indicate that the current system status is Normal and Candidate, respectively, are proposed to represent the system does not have service that can be exploited by worm; otherwise, it may be used for entry of worm intrusion. The system in Candidate state is moved to Vulnerability state when security center announces that some vulnerable alerts appear and systematic administrator is unable to mend in time. If administrators can patch loophole before infected by worm then status will be returned from V state to C state, else the status in V state will be moved into D state. It means that system is in Damage state and infected by malicious worm. In order to move to normal state, administrators must remove the malicious worm and patch the loophole in the system to avoid infecting another machine.

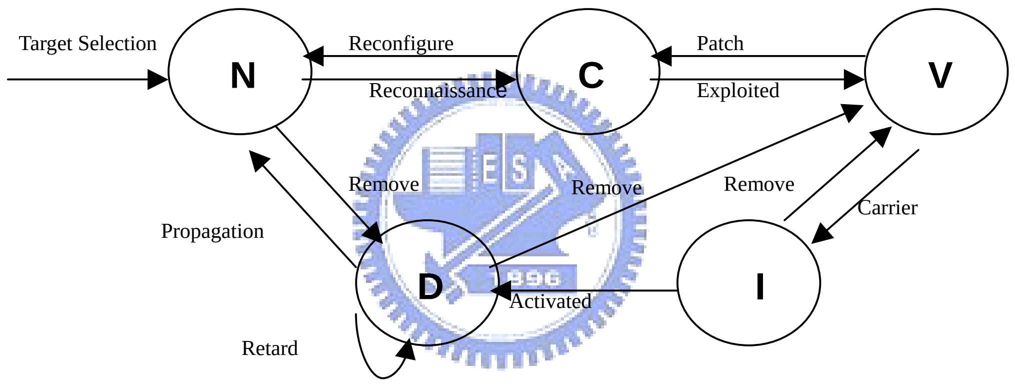


Figure 3.1: System status transition

We regard the system transition as a behavior; it can be divided into attack and defense transition. For these classes of transition we will refer to [14] in 2003 and modify it. The details will be shown as follows:

Attack transition includes target selection, reconnaissance, exploited, carrier, activated and propagation.

Attack_1) Target selection: Target selection means look for whether victims exist or not. There are a number of techniques by which a worm can discover new machines to exploit: random scanning, pre-generated target lists, internal target lists, external target lists, and passive. [14]

Scanning: Scanning entails probing a set of addresses to identify vulnerable hosts. Two simple forms of scanning are sequential (working through an address block using an ordered set of addresses) and random (trying addresses out of a block in a pseudo-random fashion).

Pre-generated target lists: An attacker could obtain a target list in advance, creating a "hit-list" of probable victims.

Internal target lists: Many applications contain information about other hosts providing vulnerable services.

External target lists: An external target list is one, which is maintained by a separate server, such as a matchmaking service's metaserver. (A metaserver keeps a list of all the servers, which are currently active. For example, the Gamespy service maintains a list of servers for several different games.)

Passive: A passive worm does not seek out victim machines. Instead, they either wait for potential victims to contact the worm or rely on user behavior to discover new targets.

Attack_2) Reconnaissance: Scanning service that is victims offered usually uses technique of port scanning. Service includes client, server and client-server.

Client: Client programs request service from a server by sending it a message such as Netscape or Internet Explorer. Referring back to the Web example, a Web browser is a client we use everyday to request Web pages. For example, when you clicked the link to read this article, your browser sent a message to a Web server. In response, your browser received the html page you are now reading. A Web browser represents many client programs, which manage the graphical user interface (GUI) or display portion of an application; determining the presentation of the service provided by an application.

Server: A server is used to manage and provide special services like IIS or Apache. Servers are generally passive as they wait for a client request. During these waiting periods servers can perform other tasks or perform maintenance. Unlike the client, the server must continually run because clients can request service at any time. Clients on the other hand only need to run when they require service.

Client-Server: Such as in peer-to-peer networks, each machine provides services and consumes services.

Attack_3) Exploited: Use of vulnerability to violate policy. Attacks exploit tricks to attack the service that victims offered such as buffer overflow, wrong configuration, back door, etc.

Buffer overflow: A buffer overflow exploit works by feeding the program specially crafted input content that is designed to overflow the allocated data storage buffer and change the data that follows the buffer in memory.

Wrong configuration: User's mistake caused by themselves. The user keeps the thing at their peril and will be prima-facie liable for the consequences of the hijacking / configuration (inadequate security) and any damage which arises as a natural consequence of this.

Back door: Back door (or "trap door", "wormhole"). A hole in the security of a system deliberately left in place by designers or maintainers. The motivation for such holes is not always sinister; some operating systems, for example, come out of the box with privileged accounts intended for use by field service technicians or the vendor's maintenance programmers.

Attack_4) Carrier: The means by which propagation occurs can also affect the speed and stealth of a worm including self-carried, second channel, and embedded. [14]

Self-Carried: A self-carried worm actively transmits itself as part of the infection process.

Second Channel: Some worms, such as Blaster, require a secondary communication channel to complete the infection. Although the exploit uses RPC, the victim machine connects back to the infecting machine using TFTP to download the worm body, completing the infection process.

Embedded: An embedded worm sends itself along as part of a normal communication channel, either appending to or replacing normal messages.

Attack_5) Activated: The means by which a worm is activated on a host also drastically affects how rapidly a worm can spread, because some worms can arrange to be activated nearly immediately whereas others may wait days or weeks to be activated. [14]

Human Direct: The slowest activation approach requires a worm to convince a local user to execute the local copy of the worm.

Human Indirect: Similarly, many worms are activated when the user performs some activity not normally related to a worm, such as resetting the machine, logging in and therefore executing login scripts, or opening a remotely infected file.

Scheduled Process: The next fastest worms activate using scheduled system processes.

Self-Activation: The worms that are fastest activated are able to initiate their own execution by exploiting vulnerabilities in services that are always on and available (e.g., Code Red exploiting IIS Web servers).

Attack_6) Propagation: After infected by worm, there are many abnormal behaviors generated including denial of service, worm maintenance, etc.

Denial of Service: DOS attack will make normal routines paralyzed such as spam-relays, Internet remote control, Internet DOS, data damage etc.

Worm Maintenance: Past worms such as W32/sonic have included a crude update mechanism: querying web sites for new code.

Defense transition includes reconfigure, patch, remove and retard.

Defense_1) Reconfigure: Disable unnecessary or vulnerable services that make the virus unable to invade.

Defense_2) Patch: If administrators can patch loophole before infected by worm, worm can still not invade system though the service utilized. Otherwise, attacker can use those weaknesses to get privilege of administrator and destroy.

Defense_3) Remove: Worm will add, delete or modify files to file system. If we want to return to normal status, we must recover the infected file and remove worms that exist in system.

Defense_4) Retard: When a new worm generated, security center may not offer the solution in time. They will slow down the abnormal flow that worm produces in the network to get more time and solve the problem.

3.2: Definition of Worm Knowledge Class

A Rule-Base can record various knowledge concepts in a specific domain and each Knowledge Class (KC) in the Rule-Base represents different concept of the domain knowledge.

Since the knowledge of worms consists lots of concepts according to worm life cycle, we model six knowledge classes of worm. A KC consists of rules, relations with other KCs and fact declarations as shown in Figure 3.2. The working model is composed of different KCs and the transferences (e.g., Trigger, Acquire, Reference, and Extension-of, etc) between KCs. The relationships between KCs are divided into two kinds-dynamic and static. The former two (e.g., Trigger and Acquire) are dynamic because they are activated conditionally in the action part of a rule, while the latter two are static.

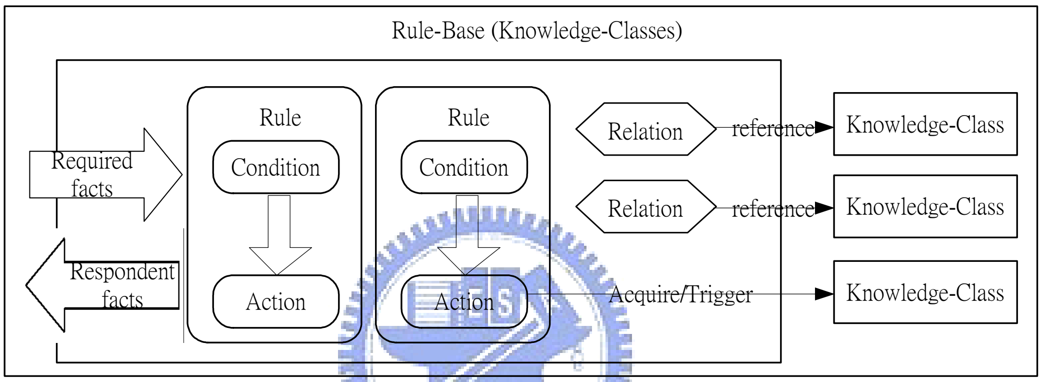


Figure 3.2: The knowledge class in a rule base

The knowledge classes can assist experts to construct knowledge base easily, and representation of worm by knowledge class can make knowledge engineer or domain expert more understandable, maintainable and reusable. These six knowledge classes list as follows:

KC_Basic information: This knowledge class includes discovery date, target selection, operating system etc; such as identity card that can know the brief information of worm.

KC_Service: Often worm will scan ports of the victim. The main purpose is to find out whether the victim had some service that may be invaded by the worm. Usually KC_Service can be divided into client, server and client-server.

KC_Exploitation: Worm will exploit some techniques such as buffer overflow, wrong configuration, backdoor to attack normal service to get privilege in order to reproduce. In other words, if some service has loophole then the worm can invade the user's machine by this way.

KC_Carrier: This knowledge class describes how to carry the worm body into victim machine. Therefore, we will check whether worm is hided in system.

KC_Symptoms: According to attackers and their motivation will cause different propagations and symptoms. This class describes that the abnormal behavior appearing after being infected.

KC_Defense instruction: Last knowledge class describes how to recover system to the normal status and make system become stronger.

Figure 3.3 shows the relationship of knowledge classes. As mentioned in Section 2.4, distinct system status will refer to different knowledge classes. We list the profile of knowledge classes below.

- If KC_service=false then status=N
- If KC_service=true then Trigger KC_exploitation
- If KC_service=true and KC_exploitation=false then status=C
- If KC_service=true and KC_exploitation=true and KC_carrier=false then status=V and Trigger KC_defense instruction
- If KC_carrier=true then Trigger KC_symptoms
- If KC_carrier=true and KC_symptoms=false then status=I and Trigger KC_defense instruction
- If KC_carrier=true and KC_symptoms=true then status= D and Trigger KC_defense instruction

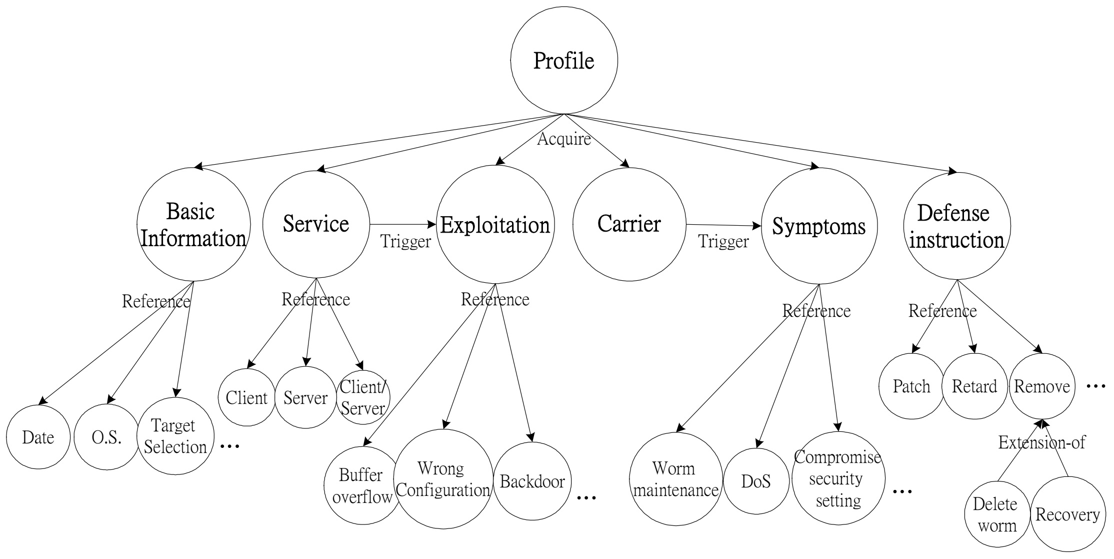


Figure 3.3: Relationship of knowledge classes

3.3: The Overview of WISE system Architecture

In order to reach the deficiency of general antivirus softwares described above, we build a worm immune service expert system. The architecture is shown in Figure 3.4. The WISE system architecture consists of three parts.

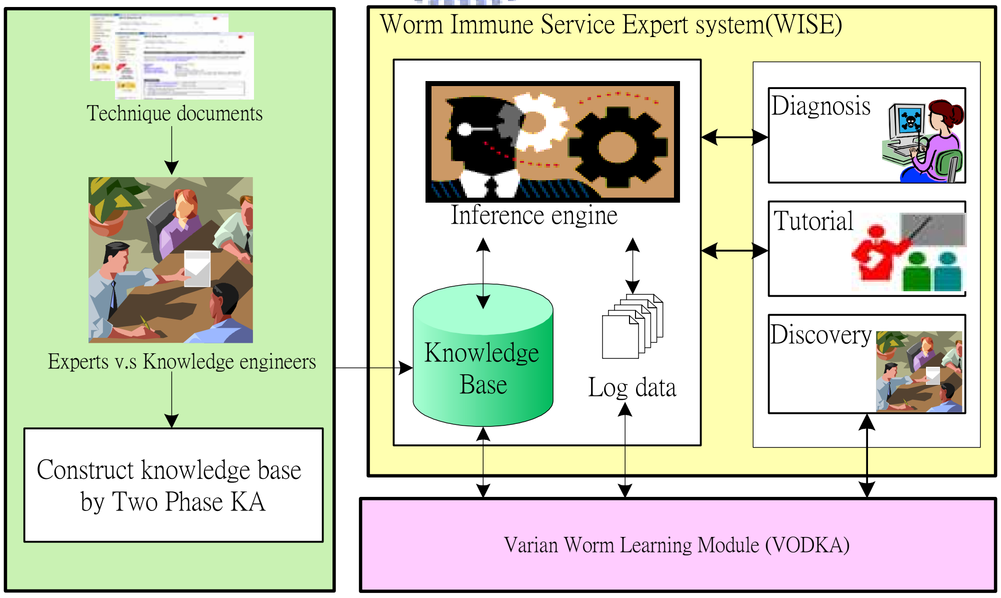

Technique documents
Experts v.s Knowledge engineers
Construct knowledge base by Two Phase KA
Worm Immune Service Expert system(WISE)
Inference engine
Knowledge Base
Log data
Diagnosis
Tutorial
Discovery
Varian Worm Learning Module (VODKA)

Figure 3.4: Architecture of WISE

Part 1: Construct knowledge base by Two Phase Knowledge Acquisition

We can get the worm domain know-how from Internet. These technological documents are non-structured, so we cannot elicit knowledge by text mining. As mentioned above, the concept of worm can be built based on the acquisition of knowledge. Two Phase Knowledge Acquisition [1] (*Tp-KA*) acquires the knowledge hierarchy and constructs the hierarchical grids in the first phase, and then transforms the knowledge hierarchy into maintainable and traceable format and extracts guiding rules from the hierarchical grids in the second phase. The detail will be introduced in Chapter 4.

Part 2: Apply learning module to learn new variant worms

Now, new variant worms are evolving too fast to keep up by experts or security center. Hence, how to collect sufficient relevant information to help experts notice the occurrence of variants and reuse the original rule base becomes one of important issues in knowledge acquisition field. We apply a new knowledge acquisition methodology, *Variant Objects Discovering Knowledge Acquisition* (*VODKA*) methodology, where each iteration including three stages (Log Collecting Stage, Knowledge Learning Stage, and Knowledge Polishing Stage) to iteratively discover the variants derived from original worm.

Part 3: Build the environment of WISE with users and domain experts

When worm infects the system, the users perceive some abnormal behaviors such as reboot of operating system, abnormal function of application program etc. These symptoms perhaps can utilize the probe to find out, but not all. The users usually will usually input what he or she looks to the search engine on the network and looks for the solution. Such a way consume large time look for correct solutions from the disorderly and unsystematic web pages. Consequently, we provide a web interface to interact to the user and also collect information of symptoms what the user cannot see by the probe.

As mentioned above, WISE is proposed to the persons of two kinds mainly. One is the user who might be infected by worm another one is the administrator of security-related organizations.

If worm infects system then the user will scan host computer by general antivirus softwares or look for help from the network. WISE can collect information that detected by general antivirus softwares and found by the user. After inference engine of *DRAMA* [16] inference, the result diagnosed will be passed to the user and teach them how to remove worm and recover to normal state. Hence, the user can obtain enough knowledge about the worm and avoid following the same old catastrophic road.

Besides the biodiversity of the natural through gene mutation, most of intrusion behaviors become more complicated because of combining several signatures of previous intrusions. However, the lack of enough information about variants may result in the difficulty of observing the occurrence of new variants for human experts. Hence, how to collect sufficient relevant information to help experts notice the occurrence of variants and reuse the original rule base becomes one of important issues. Therefore, we will use a new iterative knowledge acquisition methodology, *Variant Objects Discovering Knowledge Acquisition* (*VODKA*) methodology to assist domain experts in discovering new variant worm.

Therefore, this part contains three modules: diagnosis module, tutorial module and discovery module.

In diagnosis module, we can collect symptoms by probing or inputting with users. After infer by inference engine and knowledge base, WISE will diagnose whether worm infects host or not or whether worm can invade host or not.

In tutorial module, we will tell users how to recover system. Therefore, users can not only remove the threat of worm but also learn the knowledge of worm.

In discovery module, we will assist domain experts to discover new variant worm through inference log and refine knowledge base quickly.

Chapter 4

Worm Knowledge Base Construction

Since the knowledge is rapidly growing, many knowledge acquisition methods have been proposed to capture domain knowledge. The goal of knowledge acquisition is to elicit expertise from domain experts. One of knowledge acquisition methods is that knowledge engineers acquire the useful knowledge by interviewing with domain experts. Knowledge engineers must have enough background of the domain knowledge; otherwise it may misunderstand what domain experts are talking about. Another knowledge acquisition method is the machine learning. It will learn out some rules automatically, but it needs some available training cases. The knowledge acquisition system solves the problem of communication between domain experts and knowledge engineers without the required training cases.

We can get much worm information from technique documents in Internet, but these documents mostly are non-structured and therefore we cannot analyze it with methodology of text mining. In this thesis, we use a knowledge acquisition methodology to acquire worm knowledge and construct a worm knowledge base to assist us in diagnosing system status.

4.1: Two Phase Knowledge Acquisition

Two-phase Knowledge Acquisition (*Tp-KA*) [4] is proposed to help teachers acquire the knowledge hierarchy and the relevant teaching strategy from experts systematically and effectively. Tp-KA is also useful in other domains, which exist the knowledge hierarchy and rule-based knowledge. Therefore, we use Tp-KA to construct worm knowledge base.

The first phase is to construct the knowledge hierarchy and the hierarchical grids incrementally. The second phase is to extract the rules with meaning-embedded [9] from the hierarchy grids. The algorithm is shown as follows:

Algorithm 4.1: Worm Knowledge Base Construction Algorithm

Input: The worm domain know-how.

Output: The knowledge base of the guiding rules.

- **Phase1:**

Step1: Execute ***Knowledge Hierarchy Construction Algorithm*** to construct the knowledge hierarchy.

Step2: Execute ***Concept Hierarchy to Repertory Grids Algorithm*** to construct and fill up the hierarchical repertory grids according to the knowledge hierarchy of each worm constructed in Step1.

Step3: Execute ***Hierarchical Grids Adjustment Algorithm*** to construct the hierarchical repertory grids according to hierarchical grids combined in Step2.

- **Phase2:**

Step1: Execute ***Embedded Meaning Capturing and Uncertainty Deciding Algorithm*** (***EMCUD***) to extract guiding rules from the hierarchical repertory grids.

Step2: Store the ontological rule and the meaning-embedded rules into knowledge base.

In Phase1, three corresponding algorithms, Knowledge Hierarchy Construction Algorithm, Concept Hierarchy to Repertory Grids Algorithm, and Hierarchical Grids Adjustment Algorithm are described in part 1, Section part 2, and Section part 3, respectively. In Section part 4 describes Embedded Meaning Capturing and Uncertainty Deciding Algorithm.

Part 1: Knowledge Hierarchy Construction

To construct the concept hierarchy for worm domains, cooperation between domain experts and knowledge engineers is required. The following Knowledge

Hierarchy Construction Algorithm (KHC) can guide the constructing process of knowledge hierarchy.

Algorithm 4.2: Knowledge Hierarchy Construction Algorithm

```
Input: The worm domain know-how and skeletal of worm.
Output: The concept hierarchy format of worm.
Step1: List all elementary knowledge objects according to technical documents.
Step2: While the knowledge hierarchy is not completed, ask the domain experts
       to proceed the following sub-steps:
    Step2.1: List the meta-knowledge objects based on some of built knowledge
             objects.
    Step2.2: Establish the directed links among the meta-knowledge objects and
             the built knowledge objects.
Step3: Visit all knowledge hierarchy.
Step4: If the knowledge hierarchy belong to one of worm's skeletal, then go to
       Step4.1, else go to Step4.2.
    Step4.1: Establish the directed links among the worm's skeletal and the
             knowledge hierarchy.
    Step4.2: Build new skeletal object about the knowledge hierarchy.
Step5: If there exists any unvisited node, then go to Step3.
```

Example 4.1: Example of Knowledge Hierarchy Construction

In this example, we want to construct the concept hierarchy of Nimda. According to KHC Algorithm, domain experts extract knowledge objects after assimilating technologic documents of worm**Knowledge objects of Nimda**

Discovery date: 2001/09
Worm Length: 57,344 bytes
Target selection: Scanning
OS: Windows98
OS: WindowsMe
OS: Windows2000
OS: WindowsNT
OS: WindowsXP
IIS Exploitation: Web traversal
IE Exploitation: MIME header Network Neighborhood: Wrong configuration
New user account: guest
New file: Load.exe; Admin.dll
Modify file:System.ini; Riched20.dll
Send email
Email_attend file: readme.exe
Word,WordPad abnormal
Treatment:removal tool:Symantec
Treatment:removal tool:Trend Micro
Prevent: patch update: MS01-020; MS01-044,…

Figure 4.1 (a) shows the elementary knowledge hierarchy from knowledge object and Figure 4.1 (b) is the skeletal of worm. After interacting to domain expert, we can build a concept hierarchy of Nimda as shown in Figure 4.1 (c).

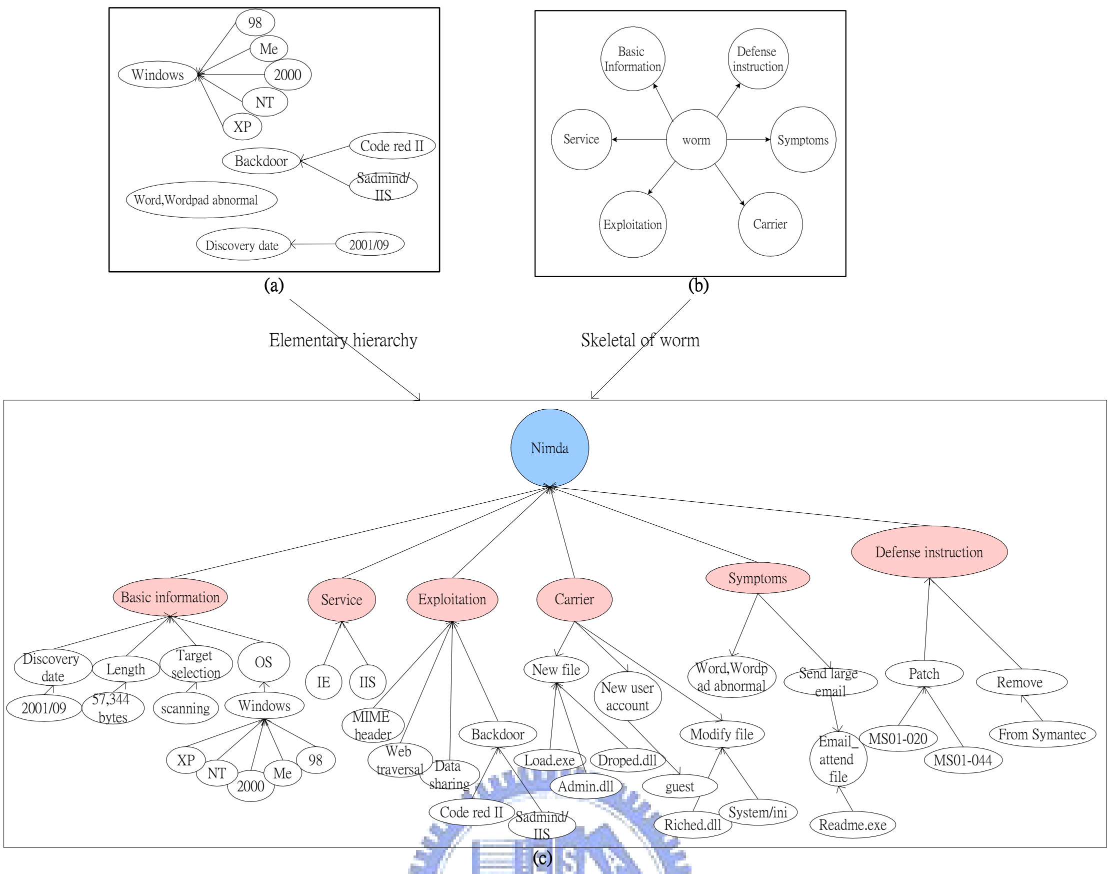


Figure 4.1: Concept hierarchy of Nimda

Part 2: Concept Hierarchy to Repertory Grids

Due to experts cannot easily understand concept hierarchy of worm intuitively; we use repertory grids to represent the knowledge. Repertory grids, based on Kelly's Personal construct theory [8], which reports how people make sense of the world, can be used as knowledge acquisition tools in the development of expert systems. The technique assists in identifying different objects in a domain and distinguishing among these objects. A single repertory grid represented as a matrix whose columns have elements object (labels) and whose rows have construct attribute (labels) can classify a class of objects, or individuals. Now, several models have been proposed for handling uncertainties in expert systems. EMYCIN [7] Certainty Factor (CF) model which was first used in the medical expert system decides the degree of belief of a rule.In the worm of concept hierarchy is shown in Figure 4.3(c), each node of the hierarchy indicates a knowledge object in accordance with characteristics of worm characteristic. The parent node can be considered as the meta-knowledge of the child node. For each knowledge object, a repertory grid is used to describe its properties, with the objects as column headers and attributes as row headers of the grid. Based upon the concept hierarchy, the hierarchical relationship between the parent grid and the child grid is determined. Thus, the following Concept Hierarchy to Repertory Grids Algorithm (CH2RD) is proposed to construct and fill up the hierarchical multiple grids according to the knowledge concept hierarchy built in the first stage of the first phase in the cooperation of experts with knowledge engineers. The algorithm is shown as follows:

Algorithm 4.3: Concept Hierarchy to Repertory Grids Algorithm

Input: The worm map in concept hierarchy format.

Output: The hierarchy repertory grids.

Step1: Visit the hierarchy according to *Depth First Search* algorithm

Step2:IF this node's is left node then go to **Step3**, else go to **Step6**.

Step3:IF parent is skeletal then go to **Step4**, else go to **Step6**.

Step4: Elicit attribute from the node value, and the pair of [attribute, worm] of the grid is equal to "True".

Step5: Go to **Step8**.

Step6: Elicit attribute from the node's parent value.

Step7: Fill in the values of all the node parent's children of the grid.

Step8: If there exists any unvisited node, then go to **Step1**.

Step9: Stop.

Example 4.2: Example of Concept Hierarchy to Repertory Grids

For the ontological lattice shown in Figure 4.1(c), CH2RD Algorithm constructs the following grids.

The **Nimda** grid shows in Table 4.1. It contains six attributes: Basic information, Service, Exploitation, Carrier, Symptoms and Defense instruction.

Table 4.1: Table of Nimda grid

Level 1	**Nimda**
Basic information	{Discovery date; Length; Target selection; OS}
Service	{IIS; IE; Network Neighborhood}
Exploitation	{MIME header; Web traversal; Wrong configuration}
Carrier	{New file; Modify file}
Symptoms	{Word, WordPad abnormal; Send email; New user account }
Defense instruction	{Patch; Remove}

The **Basic information** grid shows in Table 4.2 contains four attributes: Discovery date, Length, Target selection and OS. This grid is descendant of **Nimda** grid by Basic information attribute.

Table 4.2: Table of Basic information grid

Level 2	**Nimda**
Discovery date	2001/09
Length	57344 bytes
Target selection	Scanning
OS	Windows

The **Service** grid shows in Table 4.3. It contains two attributes: IE, IIS and

Network Neighborhood. This grid is descendent of **Nimda** grid by Service attribute.

Table 4.3: Table of Service grid

Level 2	**Nimda**
IE	True
IIS	True
Network Neighborhood	True

The **Exploitation** grid shows in Table 4.4. It contains four attributes: MIME header, Web traversal and Wrong configuration. This grid is descendant of **Nimda** grid by Exploitation attribute.

Table 4.4: Table of Exploitation grid

Level 2	**Nimda**
MIME header	True
Web traversal	True
Wrong configuration	True

The **Carrier** grid shows in Table 4.5. It contains three attributes: New file and Modify file. This grid is descendant of **Nimda** grid by Carrier attribute.

Table 4.5: Table of Carrier grid

Level 2	**Nimda**
New file	{Admin.dll; Load.exe; Droped.dll}
Modify file	{Riched.dll; System.ini}

The **Symptoms** grid shows in Table 4.6. It contains two attributes: Word, Word Pad abnormal, Send email and New user account. This grid is descendant of **Nimda** grid by Symptoms attribute.

Table 4.6: Table of Symptoms grid

Level 2	**Nimda**
Word, Word Pad abnormal	True
Send email	{Attach file}
New user account	Guest

Part 3: Hierarchical Grids Adjustment

With the growth of time, there is more and more worm too. We can construct repertory grids of each worm through CH2RD Algorithm, but each worm has its own characteristic and constructing concept hierarchy by KHC Algorithm is not the same. After merging repertory grids of every worm in each knowledge class, we find grids exist some relation between attributes. Therefore, in order to avoid conflicting rule, we purpose a Hierarchical Grids Adjustment Algorithm (HGA) to split repertory grid into hierarchical grid of incremental knowledge of worm.The relation between attributes includes Sibling Relation, Parent-child Relation and Ancestor-descendent Relation. When expert finds Sibling Relation between attributes, a parent shall be created. That is to say, expert can use a high level attribute to describe these child attributes and make knowledge easier to maintain. If there are Parent-child Relation or Ancestor-descendent Relation between attributes, we shall split repertory grid into hierarchical grids to avoid redundant inference. Briefly, two different layers of attributes are in the same layer and then shall be split into different grids and generate meta-rule to describe their relationship. The algorithm is shown as follows:

Algorithm 4.4: Hierarchical Grids Adjustment Algorithm

Input: The repertory grids.

Output: The hierarchy repertory grids.

Step1: While the hierarchy grid is not completed, ask the domain experts to proceed the following steps2-3:

Step2: Get a pair of attribute (A, B) to ask expert whether it exist the following relation.

Sibling Relation: If A and B can combine then go to **Step2.1**.

Step2.1: Elicit the parent attribute by expert and the new pair of [parent attribute, worm] of the grid is equal to "{A ∪ B ∪ Parent. Values}".

Step2.2: Push down the pair of [A, worm] and [B, worm] to lower grid.

Parent-child Relation: If B belongs with A's subset then go to **Step2.3**.

Step2.3: The pair of [A, worm] of the grid is equal to "{B ∪ A. Values}".

Step2.4: Push down the pair of [B, worm] to lower grid.

Ancestor-descendent Relation: If B belongs with attribute A's value that is C subset then go to **Step2.5**.

Step2.5: The pair of [C, worm] of the grid is equal to "{B ∪ C. Values}".

Step2.6: Push down the pair of [B, worm] lower than [C, worm] grid.

Step3: If there exists any unvisited attribute pair, and then go to **Step2.**

Step4: Expert confirms the hierarchy repertory grids.

Example 4.3: Example of Hierarchical Grids Adjustment

Figure 4.2 shows the concept hierarchy of Code Red constructing by KHC Algorithm, and Table 4.7 merged with symptoms grid of Nimda and Code Red after CH2RD Algorithm.

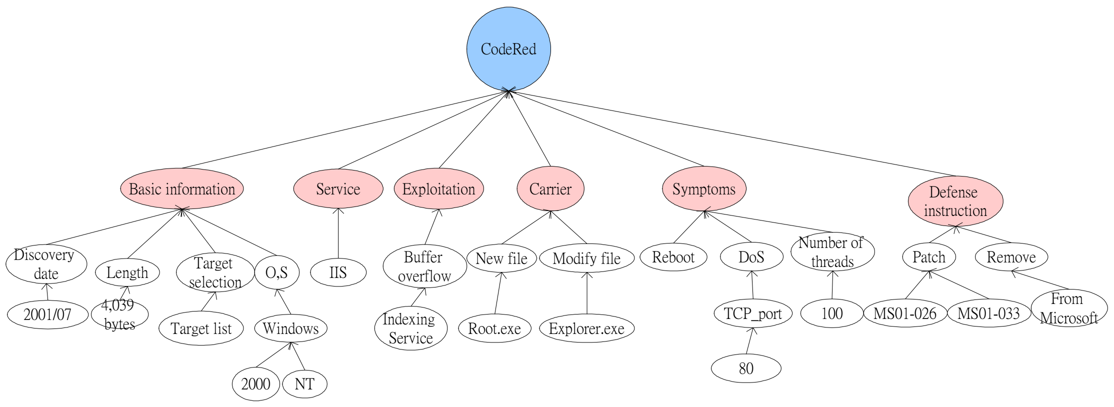


Figure 4.2: Concept hierarchy of Code Red

Table 4.7: Merging Symptoms grid before splitting

Level 2	**Nimda**	**Code Red**
Word, Word Pad abnormal	True	X
Send email	{Attach file}	X
Reboot	X	True
DoS	X	TCP_port
Number of threads	X	100

Therefore, we can find that the attribute of "Send email" is a subset of "DoS", Parent-child Relation is found by expert and original grid split into hierarchy grid in Table 4.8 through HGA Algorithm.

Table 4.8: Merging Symptoms grid after splitting

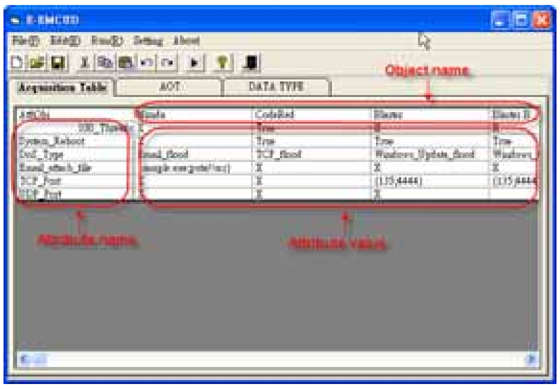
Level 2	**Nimda**	**Code Red**
Word, Word Pad abnormal	True	X
Reboot	X	True
DoS	Send email	TCP_port
Number of threads	X	100

Level 3	**Nimda**	**Code Red**
Send large email	{Attach file}	X
TCP_port	X	80

Part 4: Embedded Meaning Capturing and Uncertainty Deciding

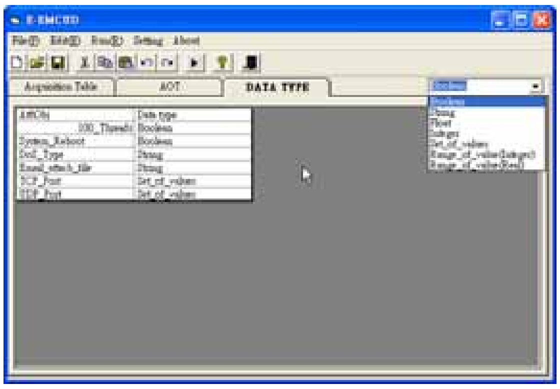
In order to make rule to generate easily, we implement EMCUD [9] to assist experts to generate original rules and embedded rules. We apply the first phase of Worm Knowledge Base Construction algorithm to derive the knowledge of worm and fill the repertory grids in acquisition table that is shown in Figure 4.3. The data types support in system are shown as follows and in Figure 4.4.

Boolean	: True or False
Single value	: an Integer, Float or String
Set of values	: a set of Integer, Float or String, e.g., {1,2,4,5}, {Admin.dll; Load.exe}.
Range of value	: a range of integers or float numbers, e.g., {1.5-3.4}, [12-18].



AttrObj	Nimda	CodeRed	Blaster	Blaster B
100_Thread		True	X	X
System_Reboot		True	True	True
DoS_Type	email_flood	TCP_flood	Windows_Update_flood	Windows_
Email_attach_file	[illegible]	X	X	X
TCP_Port		X	{135,4444}	{135,4444
UDP_Port		X	X	

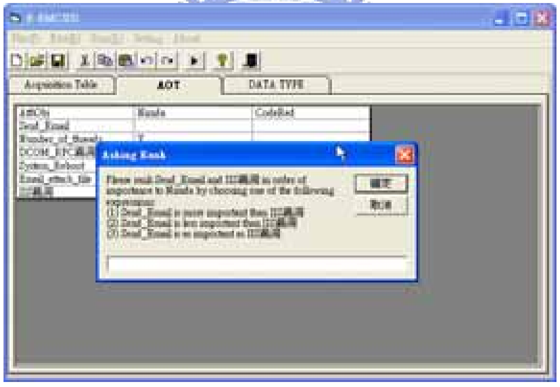
Figure 4.3:Acquisition of worm



AttrObj	Data type
100_Threads	Boolean
System_Reboot	Boolean
DoS_Type	String
Email_attach_file	String
TCP_Port	Set_of_values
UDP_Port	Set_of_values

Figure 4.4:Date type of EMCUD

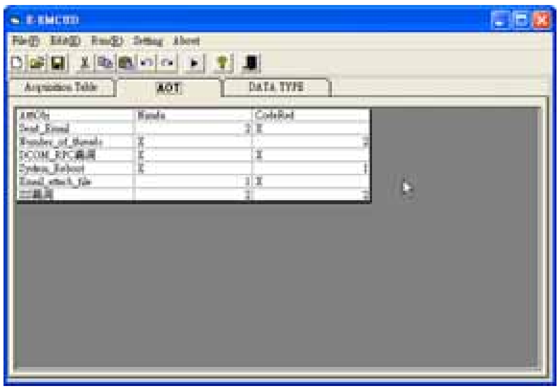
To capture the embedded meanings of the result grid, EMCUD constructs an Attribute-Ordering Table (AOT) to record the importance of each attribute to each object. Figure 4.5 shows the process of deciding the relative degrees of importance for attributes to object of each column and Figure 4.6 shows the result of AOT.



AttrObj	Nimda	CodeRed
Send_Email		
Number_of_threads	Y	
DCOM_RPC漏洞		
System_Reboot		
Email_attach_file		
IIS漏洞		

Please rank Send_Email and IIS漏洞 in order of importance to Nimda by choosing one of the following expressions:
(1) Send_Email is more important than IIS漏洞
(2) Send_Email is less important than IIS漏洞
(3) Send_Email is as important as IIS漏洞

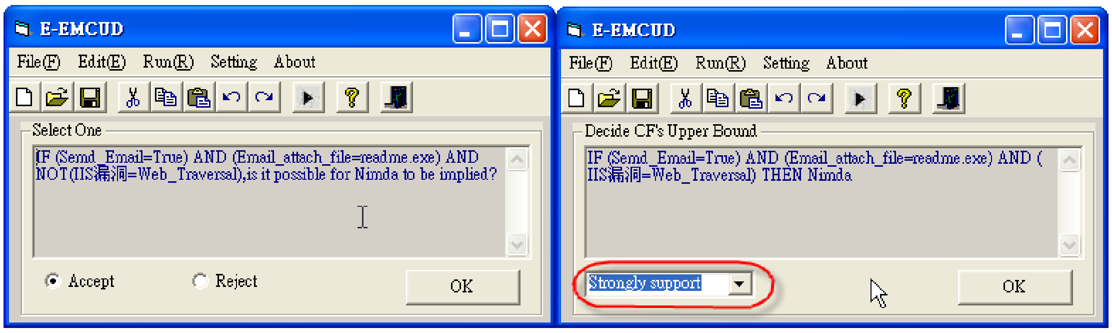
Figure 4.5: Process in construing AOT



AttrObj	Nimda	CodeRed
Send_Email	2	X
Number_of_threads	X	2
DCOM_RPC漏洞	X	X
System_Reboot	X	1
Email_attach_file	1	X
IIS漏洞	2	2

Figure 4.6: The result of AOT

To capture embedded meanings from the original rules after finishing the table of acquisition table, AOT and Data type. The process of generating embedded rules is shown in Figure 4.7.


E-EMCUD
File(F) Edit(E) Run(R) Setting About
Select One
Accept
Reject
OK
E-EMCUD
File(F) Edit(E) Run(R) Setting About
Decide CF's Upper Bound
OK

```
IF (Semd_Email=True) AND (Email_attach_file=readme.exe) AND
NOT(IIS漏洞=Web_Traversal),is it possible for Nimda to be implied?
```

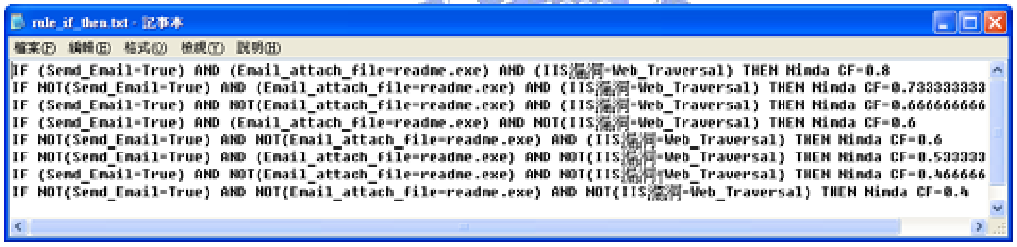
[✓ Accept] Reject

```
IF (Semd_Email=True) AND (Email_attach_file=readme.exe) AND (
IIS漏洞=Web_Traversal) THEN Nimda
```

Strongly support

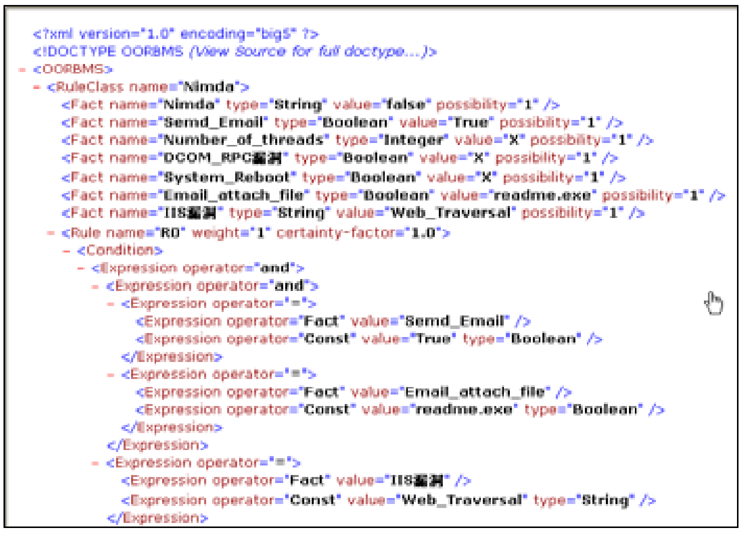
Figure 4.7: The process in generating embedded rules

After running of EMCUD, we can get the original and embedded rules of each object with Certainty Factor (CF) in Figure 4.8. At the same time we will generate a XML file that records the rules generated by EMCUD is shown in Figure 4.9. This file used for knowledge base of DRAMA [16].


rule_if_then.txt - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)

```
IF (Send_Email=True) AND (Email_attach_file=readme.exe) AND (IIS漏洞=Web_Traversal) THEN Nimda CF=0.8
IF NOT(Send_Email=True) AND (Email_attach_file=readme.exe) AND (IIS漏洞=Web_Traversal) THEN Nimda CF=0.733333333
IF (Send_Email=True) AND NOT(Email_attach_file=readme.exe) AND (IIS漏洞=Web_Traversal) THEN Nimda CF=0.666666666
IF (Send_Email=True) AND (Email_attach_file=readme.exe) AND NOT(IIS漏洞=Web_Traversal) THEN Nimda CF=0.6
IF NOT(Send_Email=True) AND NOT(Email_attach_file=readme.exe) AND (IIS漏洞=Web_Traversal) THEN Nimda CF=0.6
IF NOT(Send_Email=True) AND (Email_attach_file=readme.exe) AND NOT(IIS漏洞=Web_Traversal) THEN Nimda CF=0.533333
IF (Send_Email=True) AND NOT(Email_attach_file=readme.exe) AND NOT(IIS漏洞=Web_Traversal) THEN Nimda CF=0.466666
IF NOT(Send_Email=True) AND NOT(Email_attach_file=readme.exe) AND NOT(IIS漏洞=Web_Traversal) THEN Nimda CF=0.4
```

Figure 4.8: The result of generated rules



```
  <?xml version="1.0" encoding="big5" ?>
  <!DOCTYPE OORBMS (View Source for full doctype...)>
- <OORBMS>
  - <RuleClass name="Nimda">
      <Fact name="Nimda" type="String" value="false" possibility="1" />
      <Fact name="Semd_Email" type="Boolean" value="True" possibility="1" />
      <Fact name="Number_of_threads" type="Integer" value="X" possibility="1" />
      <Fact name="DCOM_RPC漏洞" type="Boolean" value="X" possibility="1" />
      <Fact name="System_Reboot" type="Boolean" value="X" possibility="1" />
      <Fact name="Email_attach_file" type="Boolean" value="readme.exe" possibility="1" />
      <Fact name="IIS漏洞" type="String" value="Web_Traversal" possibility="1" />
    - <Rule name="R0" weight="1" certainty-factor="1.0">
      - <Condition>
        - <Expression operator="and">
          - <Expression operator="and">
            - <Expression operator="=">
                <Expression operator="Fact" value="Semd_Email" />
                <Expression operator="Const" value="True" type="Boolean" />
              </Expression>
            - <Expression operator="=">
                <Expression operator="Fact" value="Email_attach_file" />
                <Expression operator="Const" value="readme.exe" type="Boolean" />
              </Expression>
            </Expression>
          - <Expression operator="=">
              <Expression operator="Fact" value="IIS漏洞" />
              <Expression operator="Const" value="Web_Traversal" type="String" />
            </Expression>
```

Figure 4.9: An XML file of rules

4.2: Variant Worm Learning Module

Since the embedded rules with diverse CF values represent the different supports to classify objects, some candidate of a new variant might trigger the ones with marginally acceptable CF. In order to analyze the behaviors of these embedded rules, An iterative Variant Objects Discovering Knowledge Acquisition (VODKA) [15] framework in which each iteration consists of three stages: Log Collecting Stage, Knowledge Learning Stage, and Knowledge Polishing Stage. Initially, the embedded rule base will be created according to the original main acquisition table using *EMCUD* or *VODKA*. Then in each iteration the inference behaviors (facts/attribute-value pairs/raw) will be collected to discover the candidates of the variants during Stage I according to the meta-knowledge. The attribute-value pair will be treated as an item and a set of negated attribute-value pairs will be treated as a transaction to discover the association between interesting (minor) attribute-value pairs in Stage II. Consequently, interacting with domain experts through the new variants acquiring procedure could generate the new variants acquisition table based on the discovered knowledge. Finally, the rules of new variants will be incrementally generated and the main acquisition table will be iteratively adjusted using *E-EMCUD* to achieve the ability of grid evolution in Stage III. The algorithm of *VODKA* [15] is shown as follows.

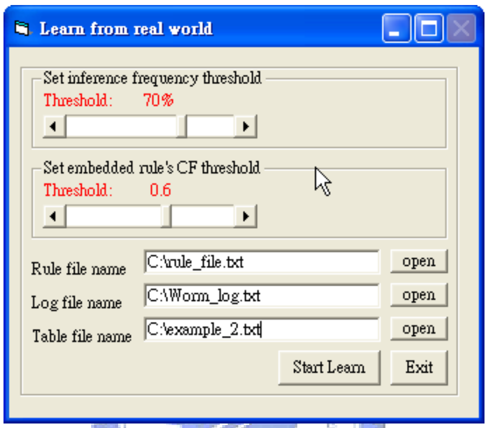
Algorithm 4.5: VODKA Algorithm

```
Input: The original main acquisition table $T$ and embedded rule base $RB$.
Output: The rules with embedded meaning about variants.
Stage I: Collect all facts of the weak embedded rules as real inference log of the $RB$.
Stage II: Generate the new variants acquisition table $T'$.
    Step1: Discover large itemsets $L$ using the inference log.
    Step2: Generate $T'$ using $L$ and additional attributes provided by experts.
Stage III: Use E-EMCUD to generate rules of new variants.
    Step1: Generate rules according to $T'$.
    Step2: Merge $T'$ into original main acquisition table $T$.
```

VODKA has been implemented based upon *DRAMA*, an object-oriented inference engine with *NORM* (*New Oriented-original Rule-based Model*) knowledge representation providing high maintainability, reusability, sharability, and abstraction for rule-based system and the *E-EMCUD* has also been implemented to refine the embedded rule base.

In Log Collecting Stage, the meta rule MR_1 is used to count the fired frequency of each embedded rule ($C_{i,j}$). The meta rule MR_2 means that all facts (attribute-value pairs) of the embedded rules with marginally acceptable CF lower than strong CF bound threshold (TH_{CF}) are logged as a record, ($R_{i,j}$, A_1, A_2, ….,A_k, CF($R_{i,j}$)). The meta rule MR_3 means that if there exists one weak embedded rule with fired frequency exceeding the warning line threshold (TH_{CNT}), new variants may be discovered using the iterative process of *VODKA*. The meta rule MR_4 means that *VODKA* will be executed periodically to refresh the new variants acquisition table. The *TimeOut* will be reset when R_{M3} or R_{M4} is triggered. Therefore, we must set the threshold and the file of rules and tables for the Knowledge Learning Stage in Figure 4.10. Figure 4.11 shows the inference log to collect by inference engine [15] .

MR_1: IF $R_{i,j}$ is fired THEN Increase $C_{i,j}$ by one.
MR_2: IF CF($R_{i,j}$) $\leq TH_{CF}$, THEN Log $R_{i,j}$.
MR_3: IF $C_{i,j} \geq$ THcnt **AND** CF($R_{i,j}$) $\leq TH_{CF}$ THEN Run **New Variants Acquiring Algorithm** to acquire the new variants acquisition table and Reset *TimeOut*.
MR_4: IF *TimeOut* = TH_{Period} Then Run **New Variants Acquiring Algorithm** and Reset *TimeOut*.


Set inference frequency threshold
Threshold: 70%

Set embedded rule's CF threshold
Threshold: 0.6

Rule file name: C:\rule_file.txt
Log file name: C:\Worm_log.txt
Table file name: C:\example_2.txt

Figure 4.10: Setting before learning

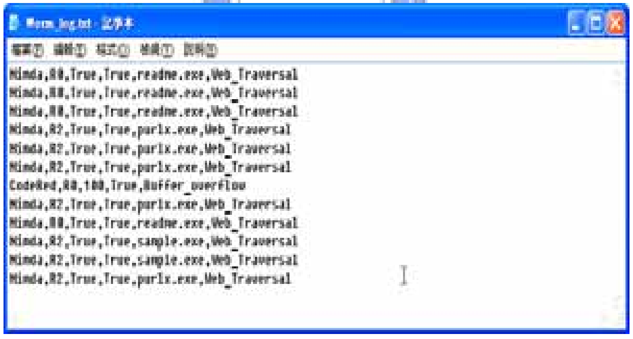
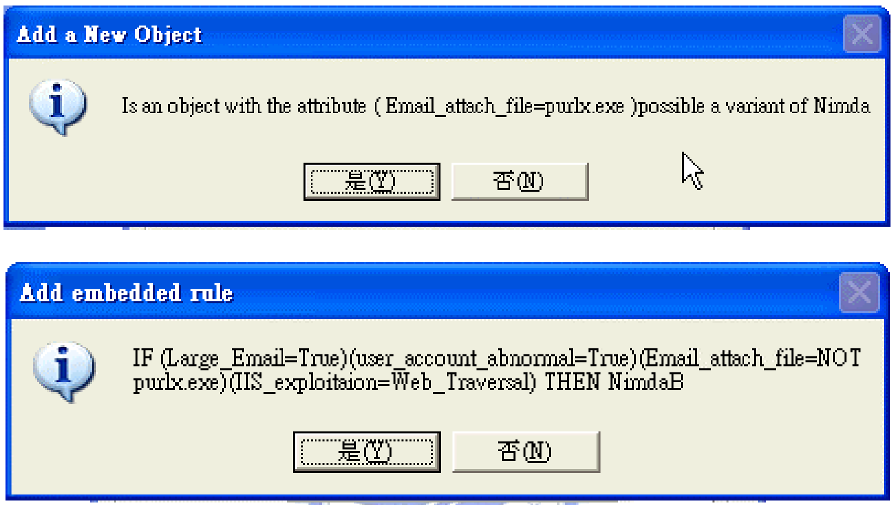

```
Nimda,R0,True,True,readme.exe,Web_Traversal
Nimda,R0,True,True,readme.exe,Web_Traversal
Nimda,R0,True,True,readme.exe,Web_Traversal
Nimda,R2,True,True,purlx.exe,Web_Traversal
Nimda,R2,True,True,purlx.exe,Web_Traversal
Nimda,R2,True,True,purlx.exe,Web_Traversal
CodeRed,R0,100,True,Buffer_overflow
Nimda,R2,True,True,purlx.exe,Web_Traversal
Nimda,R0,True,True,readme.exe,Web_Traversal
Nimda,R2,True,True,sample.exe,Web_Traversal
Nimda,R2,True,True,sample.exe,Web_Traversal
Nimda,R2,True,True,purlx.exe,Web_Traversal
```

Figure 4.11: Inference log

In the Knowledge Learning Stage, the new variants acquisition table will be generated through interacting with domain experts based upon the observation of inference log.

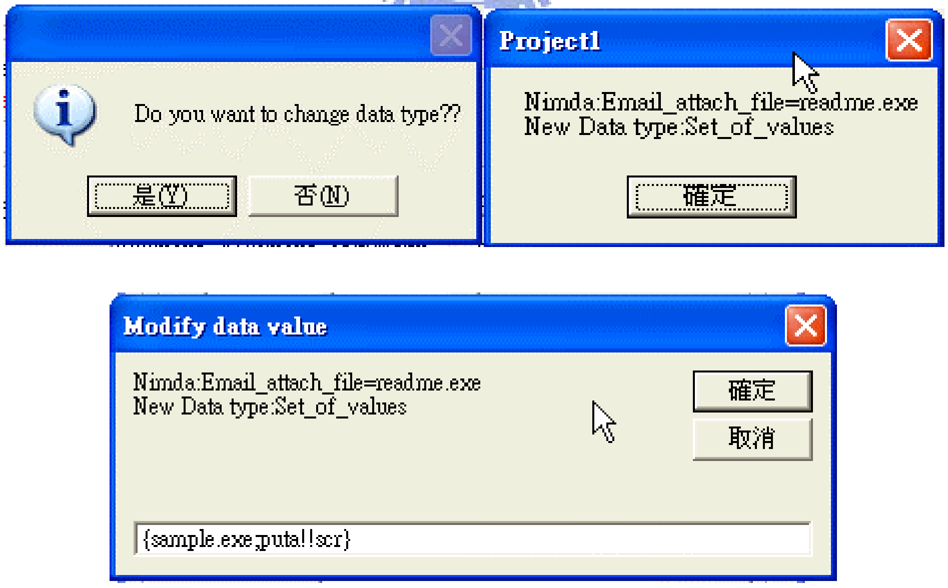
After generating the large itemsets from inference log, new variants acquisition table might be elicited by interacting with domain experts based upon the new variants acquiring algorithm. The object class using unclear attributes would be split accordingly, if the experts reconfirm the addition of the new variant object that shows in Figure 4.12(a). Thus, one of three recommendations including no change, modifying the data types of minor attributes that shows in Figure 4.12(b), adding new attributes that shows in Figure 4.12(c), will be further given to adjust the main acquisition table.



Is an object with the attribute (Email_attach_file=purlx.exe)possible a variant of Nimda

IF (Large_Email=True)(user_account_abnormal=True)(Email_attach_file=NOT purlx.exe)(IIS_exploitaion=Web_Traversal) THEN NimdaB

(a)



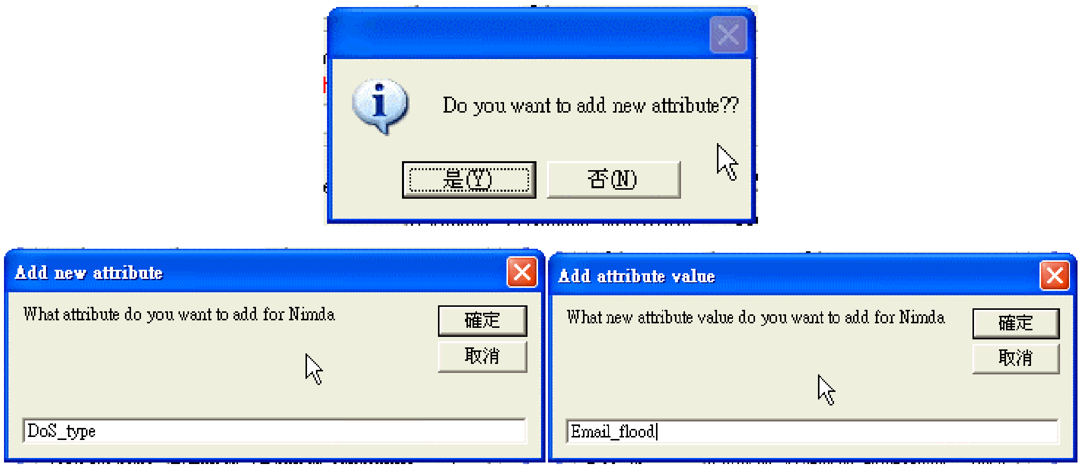
Do you want to change data type??

Nimda:Email_attach_file=readme.exe
New Data type:Set_of_values

Nimda:Email_attach_file=readme.exe
New Data type:Set_of_values

{sample.exe,puta!!scr}

(b)


Do you want to add new attribute??

是(Y) 否(N)

Add new attribute

What attribute do you want to add for Nimda

確定 取消

DoS_type

Add attribute value

What new attribute value do you want to add for Nimda

確定 取消

Email_flood

(c)

Figure 4.12: Process in Learning Stage

In Knowledge Polishing Stage, *Extended EMCUD* (*E-EMCUD*) generates new embedded rules and adjust original embedded rule base. Therefore, we can gracefully update the embedded rule base using the small new variants acquisition table instead of the whole large main acquisition table. After Knowledge Polishing Stage, system of VODKA will also generate an XML file of adjusting from original knowledge base.

Chapter 5

Implementation and Evaluations

At the time of the writing: (1) the main operating system deployed is Microsoft Windows XP; (2) the expert system tool is DRAMA enterprise 2.5; (3) the web server packages deployed is Apache Tomcat 4.1; (4) the implementation of VODKA is in Microsoft Visual Basic 6.0. Interested users could refer to the web site (e.g., http://140.113.87.155/WISE/index.jsp) for further details.

Users can connect the site to get advices on worm immune service expert system. At present, worm diagnosis module, tutorial module and learning module are available. Learning module has been introduced in Chapter 4. In the following, we will mainly focus on the worm diagnosis module and tutorial module. Figure 5.1 shows the user interface of WISE.

In Section 5.1 and Section 5.2, the implementation of diagnosis module and tutorial module, are described, respectively. The evaluation of WISE will be presented in Section 5.3.

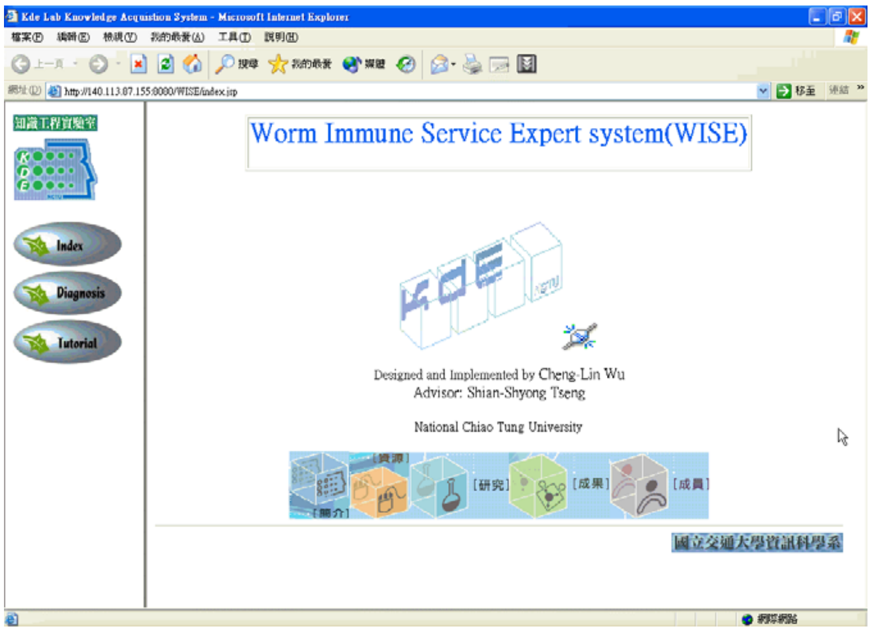

Kde Lab Knowledge Acquistion System - Microsoft Internet Explorer
http://140.113.87.155:8000/WISE/index.jsp
知識工程實驗室
Index
Diagnosis
Tutorial
Worm Immune Service Expert system(WISE)
Designed and Implemented by Cheng-Lin Wu
Advisor: Shian-Shyong Tseng
National Chiao Tung University
[資源]
[研究]
[成果]
[成員]
[簡介]
國立交通大學資訊科學系

Figure 5.1: Interface of WISE

5.1: Implementation of Diagnosis Module

When the users perceive that worms may infect system, they will look for answers from the Search Engine, e.g., Google. Although the users will find the solutions eventually, they will also spend much time to find these relevant results due to lack of domain knowledge. Therefore, we provide a diagnosis module to assist the users to solve the problems.

In diagnosis module, we will diagnose through bottom-up method. In other words, WISE will make use of system profile or abnormal behavior to judge whether could be invaded or infected by worm. First, we must collect the symptoms of user machine that can be gained from general antivirus products detected, probe or users perceived. According to these symptoms from clients, WISE can provide the appropriate solutions based on the inferred result in OORB. The solutions of WISE contain reconfiguration, patch and removal. If the worms have not invaded the existing loopholes in the user system, then defense instruction of reconfiguration or patch will be applied. Therefore, we can reach the preventing in advance. In another case, if the worms have invaded the existing loopholes in the user system, we will give advise of reconfiguration, patch and removal. Otherwise, if worms have infected the user system and the existing loopholes have been patched later, then defense instruction of removal will be only suggested. So the users can learn how to defense these malicious attacks.

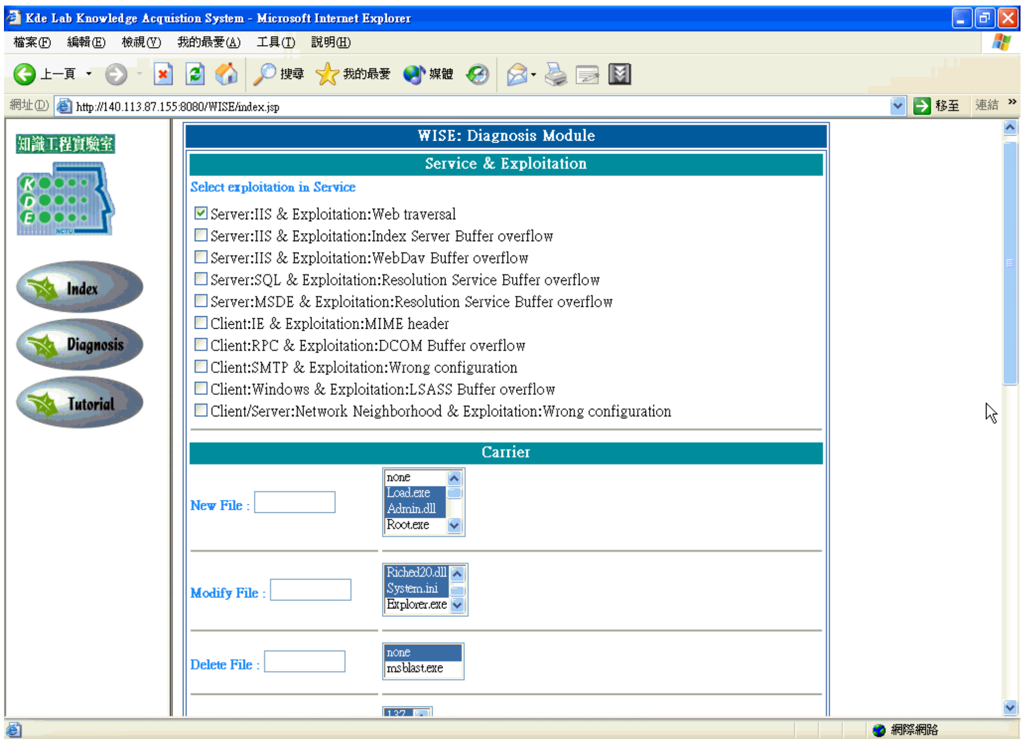
5.2: Implementation of Tutorial Module

When general antivirus products probe that worms infect machine, those products will automate to remove worms. Therefore, users of infected machine only know that system may be safe but know nothing about knowledge of worm. WISE will provide the relevant domain knowledge of worms that infect user machine and tell users how to recover from infected status. So users not only remove the threat of worms but also learn the worm at the same time.

In tutorial module, we will diagnose through top-down method. In other word, WISE will list the information of worm, service, exploitation, carrier, symptoms and defense instruction of worm that had been introduced in Chapter 3 and the detail contents of each item will be listed next. WISE will use tree structure to represent the characteristics of each worm and thereby, users can understand each worm easily

Example 5.1: Diagnosis Example of Nimda

We use Nimda as an example and capture the symptoms of Nimda that contain exploitation of IIS Web traversal, new files with Load.exe and Admin.dll, modify files of Riched20.dll and System.ini, communication port in 137 and 445, send large email with attach file of Readme.exe, new user account of Guest and abnormal application program in user machine. Those symptoms shows in Figure 5.2. After inferring in inference engine of DRAMA, Figure 5.3 shows the diagnosis result and the suitable defense instruction.



Kde Lab Knowledge Acquistion System - Microsoft Internet Explorer
http://140.113.87.155:8080/WISE/index.jsp
Index
Diagnosis
Tutorial

WISE: Diagnosis Module

Service & Exploitation

Select exploitation in Service

- [✓] Server:IIS & Exploitation:Web traversal
- [ ] Server:IIS & Exploitation:Index Server Buffer overflow
- [ ] Server:IIS & Exploitation:WebDav Buffer overflow
- [ ] Server:SQL & Exploitation:Resolution Service Buffer overflow
- [ ] Server:MSDE & Exploitation:Resolution Service Buffer overflow
- [ ] Client:IE & Exploitation:MIME header
- [ ] Client:RPC & Exploitation:DCOM Buffer overflow
- [ ] Client:SMTP & Exploitation:Wrong configuration
- [ ] Client:Windows & Exploitation:LSASS Buffer overflow
- [ ] Client/Server:Network Neighborhood & Exploitation:Wrong configuration

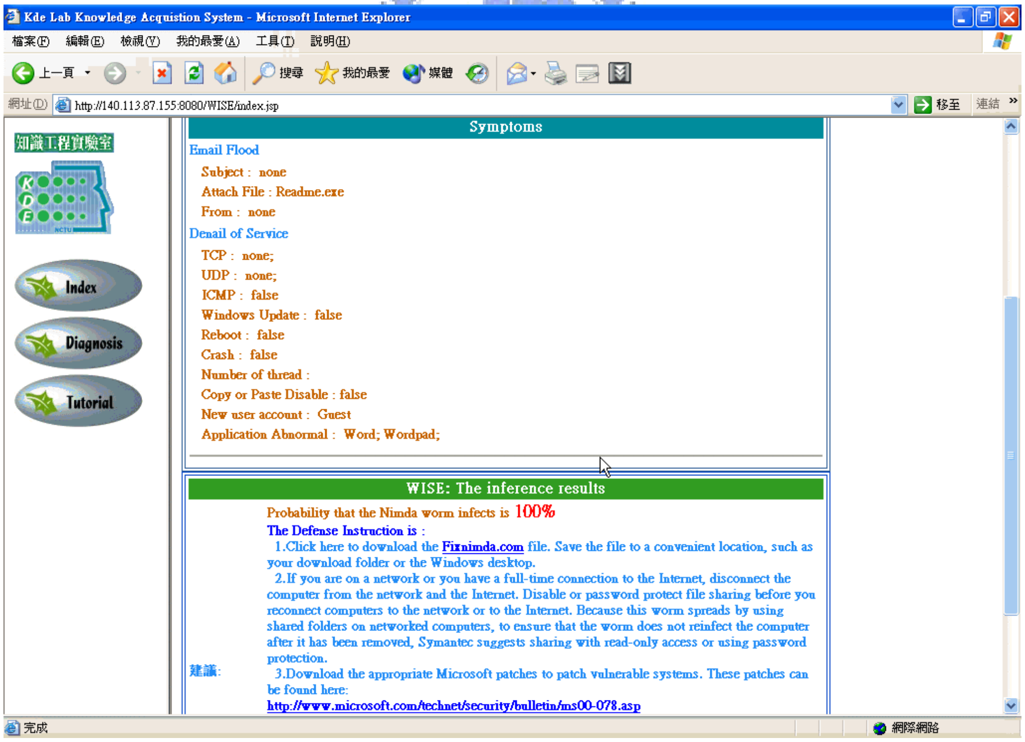
Carrier

New File : none; [✓ Load.exe]; [✓ Admin.dll]; Root.exe

Modify File : [✓ Riched20.dll]; [✓ System.ini]; Explorer.exe

Delete File : [✓ none]; msblast.exe

Figure 5.2: Symptoms of Nimda



Kde Lab Knowledge Acquistion System - Microsoft Internet Explorer
http://140.113.87.155:8080/WISE/index.jsp
Index
Diagnosis
Tutorial
完成

Symptoms

Email Flood
- Subject : none
- Attach File : Readme.exe
- From : none

Denail of Service
- TCP : none;
- UDP : none;
- ICMP : false
- Windows Update : false
- Reboot : false
- Crash : false
- Number of thread :
- Copy or Paste Disable : false
- New user account : Guest
- Application Abnormal : Word; Wordpad;

WISE: The inference results

Probability that the Nimda worm infects is 100%

The Defense Instruction is :

建議: 1.Click here to download the Fixnimda.com file. Save the file to a convenient location, such as your download folder or the Windows desktop.

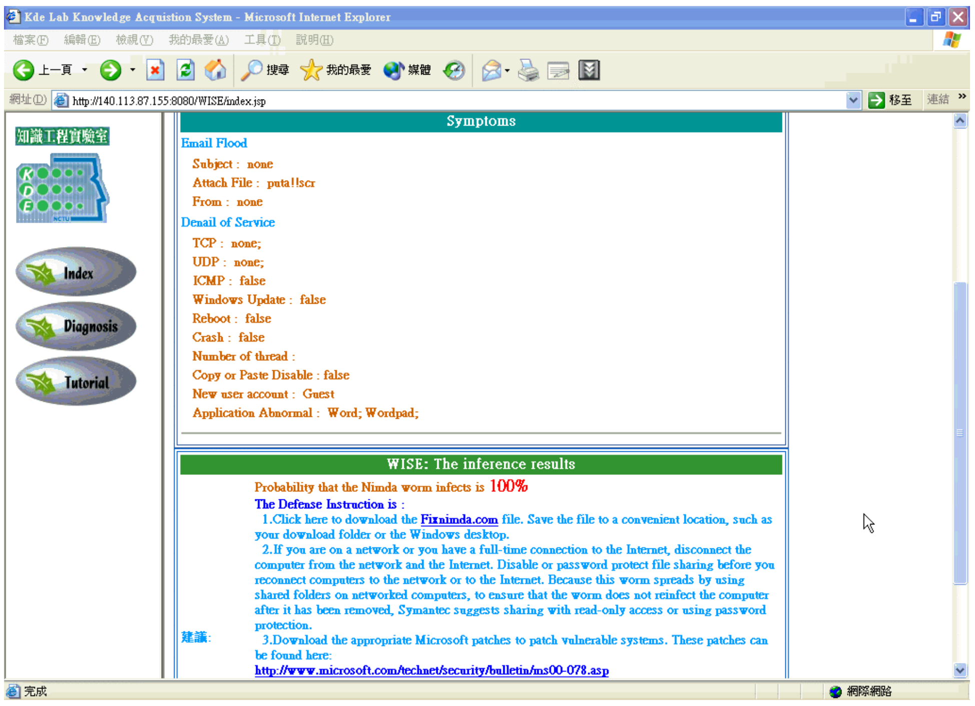
2.If you are on a network or you have a full-time connection to the Internet, disconnect the computer from the network and the Internet. Disable or password protect file sharing before you reconnect computers to the network or to the Internet. Because this worm spreads by using shared folders on networked computers, to ensure that the worm does not reinfect the computer after it has been removed, Symantec suggests sharing with read-only access or using password protection.

3.Download the appropriate Microsoft patches to patch vulnerable systems. These patches can be found here:
http://www.microsoft.com/technet/security/bulletin/ms00-078.asp

Figure 5.3: Inference result of Nimda

We know that general variant worms will modify few characteristics to avoid detecting by antivirus softwares – Nimda.B for example. The difference between Nimda and Nimda.B is email attach file but knowledge base of WISE does not include knowledge of Nimda.B now. Figure 5.4 shows the inference result of Nimda.B. We can see that WISE will tell the users the infected probability of Nimda, hence users can prevent in advance. According to knowledge class of service, exploitation, carrier and symptoms, calculation formula of probability is shown as follows and the distribution of percentage according to the destruction degree in the worm life cycle.

Probability=(service and exploitation)*20%+(carrier)*35%+ (symptoms)*45%


Kde Lab Knowledge Acquistion System - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁 搜尋 我的最愛 媒體
網址(D) http://140.113.87.155:8080/WISE/index.jsp
移至 連結
知識工程實驗室
Index
Diagnosis
Tutorial
完成
網際網路

Symptoms

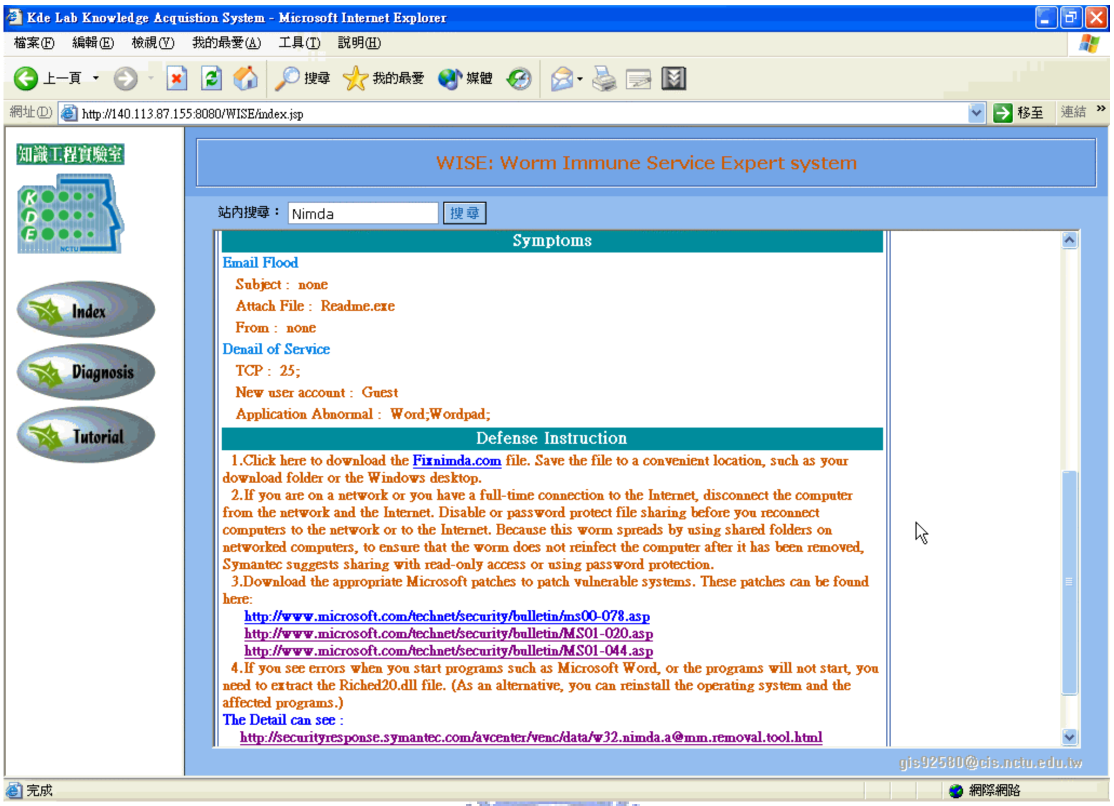
Email Flood
Subject : none
Attach File : puta!!scr
From : none
Denail of Service
TCP : none;
UDP : none;
ICMP : false
Windows Update : false
Reboot : false
Crash : false
Number of thread :
Copy or Paste Disable : false
New user account : Guest
Application Abnormal : Word; Wordpad;

WISE: The inference results

建議: Probability that the Nimda worm infects is 100%
The Defense Instruction is :
1.Click here to download the Fixnimda.com file. Save the file to a convenient location, such as your download folder or the Windows desktop.
2.If you are on a network or you have a full-time connection to the Internet, disconnect the computer from the network and the Internet. Disable or password protect file sharing before you reconnect computers to the network or to the Internet. Because this worm spreads by using shared folders on networked computers, to ensure that the worm does not reinfect the computer after it has been removed, Symantec suggests sharing with read-only access or using password protection.
3.Download the appropriate Microsoft patches to patch vulnerable systems. These patches can be found here:
http://www.microsoft.com/technet/security/bulletin/ms00-078.asp

Figure 5.4: Symptoms of Nimda.B

If we find that worms in the user system, WISE will teach the user how to kill malicious worms and recover to normal state. Besides, the users can search relevant information of worm through search engine in tutorial module to get the worms' symptoms and defense instruction.


Kde Lab Knowledge Acquistion System - Microsoft Internet Explorer
http://140.113.87.155:8080/WISE/index.jsp
Index
Diagnosis
Tutorial
gis92580@cis.nctu.edu.tw

WISE: Worm Immune Service Expert system

站內搜尋： Nimda 搜尋

Symptoms

Email Flood
Subject : none
Attach File : Readme.exe
From : none

Denail of Service
TCP : 25;
New user account : Guest
Application Abnormal : Word;Wordpad;

Defense Instruction

1.Click here to download the Fixnimda.com file. Save the file to a convenient location, such as your download folder or the Windows desktop.

2.If you are on a network or you have a full-time connection to the Internet, disconnect the computer from the network and the Internet. Disable or password protect file sharing before you reconnect computers to the network or to the Internet. Because this worm spreads by using shared folders on networked computers, to ensure that the worm does not reinfect the computer after it has been removed, Symantec suggests sharing with read-only access or using password protection.

3.Download the appropriate Microsoft patches to patch vulnerable systems. These patches can be found here:
http://www.microsoft.com/technet/security/bulletin/ms00-078.asp
http://www.microsoft.com/technet/security/bulletin/MS01-020.asp
http://www.microsoft.com/technet/security/bulletin/MS01-044.asp

4.If you see errors when you start programs such as Microsoft Word, or the programs will not start, you need to extract the Riched20.dll file. (As an alternative, you can reinstall the operating system and the affected programs.)

The Detail can see :
http://securityresponse.symantec.com/avcenter/venc/data/w32.nimda.a@mm.removal.tool.html

Figure 5.5: Defense instruction of Nimda

5.3: Evaluation

As we know, most antivirus products detect worms by file scanning or vulnerability scanning. Consequently, they lack for diagnosing in multiple views. According to the worm life cycle, different symptoms will appear in different periods. Therefore, WISE will consider including file scanning, vulnerability scanning, users views and network flows. As integrating with multiple aspects, we can clear tell the users what defense instructions should be executed. And we can also avoid human carelessness and make the result more accurate. So we can prevent in advance, besides detecting the malicious worms afterward.

Now new variant worms will modify fewer characteristics in order to escape from the antirust products detecting. This way is impractical in the WISE system because our WISE system contains embedded meaning from original worms. For this reason, we can detect those worms with less variability before the worms cause the injury to the network.

Therefore, WISE is useful for enhancing the accuracy of antivirus products and assisting the users or domain experts to find the malicious worms.

Chapter 6
Concluding Remarks

Problem of worms is very dangerous and varies greatly in Internet and the damage caused by these attacks is measured in billions of dollars. Although there are many antivirus products in the market, these products cannot guarantee to provide enough protection when unexpected worms appeared.Unlike traditional system, WISE system is a knowledge base system. It is suitable for applying in worms that evolve quickly; thereby we can refine knowledge base easily without rewriting the programs and fast prevent the infection of the virus. Besides, WISE system will integrate with multi-aspects in worm life cycle e.g., file scanning, loophole scanning, network flows, etc. and predict rationally those variant worms that have not presented by extracting the embedded meanings. Hence, we can enhance the accuracy of current commercial Antivirus products.

In this thesis, we design and implement a Worm Immune Service Expert system (WISE) to assist users and experts to discover worm easily. Our main contributions are: (1) To propose and implement a unifying framework (diagnosis, tutorial, knowledge base construction and refined, etc.) for supporting discovery of worms using web interface and expert system technology. (2) To define system status for helping detection of worms. (3) To use a Two-Phase Knowledge Acquisition algorithm for constructing the worm knowledge base.However, there are still some interesting problems that could be further discussed. To begin with, VODKA might be an interesting research topic. We need to assist experts to make a decision of new variant worm through some prior knowledge.

Bibliography

[1] Chang-Jiun Tsai, “A Study of Knowledge Management for Computer-Assisted Instruction Expert System“, 2002

[2] Chien, Eric. “Code Red Worm.” Symantec Security Response. 29 July 2003. URL:http://securityresponse.symantec.com/avcenter/venc/data/codered.worm.html. (30 July 2003)

[3] Chien, Eric. “W32.Nimda.A@mm.” Symantec Security Response. 11 July 2003. URL:http://securityresponse.symantec.com/avcenter/venc/data/w32.nimda.a@mm. html (30 July 2003) [4] C. J. Tsai and S. S. Tseng, “Building A CAL Expert System Based Upon Two-phase Knowledge Acquisition,” *Expert Systems with Applications: An Int’l Jour.*, Vol. 22, No. 3, pp. 235-248, 2002.

[5] Dan Ellis, McLean, ”Worm Anatomy and Model” ACM workshop on Rapid Malcode, 2003

[6] Dennis Eck CCNA, ”Access Control Lists to Protect a Network from Worm/DoS Attacks” GSEC Practical Assignment Version 1.4, Option 1, 2003

[7] E.H. Shortliffe and B.G. Buchanan, “A model of inexact reasoning in medicine.” Math. Bioscience, Vol. 23, pp. 351~379, 1975.

[8] G. A. Kelly, “The psychology of personal constructs.” Vol. 1 NY:W.W Norton,1955

[9] G.J. Hwang and S.S. Tseng, “EMCUD: A knowledge acquisition method which captures embedded meanings under uncertainty.” International Journal of Man-Machine Studies, Vol. 33, No. 4, pp. 431-451, 1990.

[10] Knowles, Douglas. “W32.SQLExp.Worm.” Symantec Security Response. 04 February 2003. URL:

http://securityresponse.symantec.com/avcenter/venc/data/w32.sqlexp.worm.html (29 July 2003)

[11] Knowles, Douglas et al. “W32.Blaster.Worm.” Symantec Security Response. 29 August 2003.
URL:http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.
worm.html (25 September 2003)

[12] Mark Eichin and Jon Rochlis. With Microscope and Tweezers: "An Analysis of the Internet Virus of November 1988. In IEEE Computer Society Symposium on Security and Privacy, 1989.

[13] Martin Karresand, ”Separating Trojan Horses, Viruses, and Worms- A Proposed Taxonomy of Software Weapons” IEEE Workshop, 2003.

[14] Nicholas Weaver, Vern Paxson, Stuart Staniford and Robert Cunningham, “A Taxonomy of Computer Worms” ACM workshop on Rapid Mallcode, 2003

[15] Shun-Chieh Lin, Shian-Shyong Tseng and Li-Hao Liu, VODKA: Variant Objects Discovering Knowledge Acquisition, unpublished.

[16] Y. T. Lin, S. S. Tseng, and C. F. Tsai. “Design and implementation of new object-oriented rule base management system,” *Journal of Expert Systems with Applications*, Vol. 25, Iss. 3, pp.369-385. 2003.