國立交通大學

資訊科學系

碩 士 論 文

一 個 基 於 蠕 蟲 的 合 作 式 防 禦 系 統

Collaborative Defending System for Computer Worms

研 究 生：李育松

指導教授：曾憲雄 博士

中 華 民 國 九 十 四 年 六 月

一個基於蠕蟲的合作式防禦系統

Collaborative Defending System for Computer Worms

研 究 生：李育松　　　　Student：Yu-Sung Lee

指導教授：曾憲雄　　　　Advisor：Shian-Shyong Tseng

國 立 交 通 大 學
資 訊 科 學系
碩 士 論 文

A Thesis

Submitted to Institute of Computer and Information Science

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer and Information Science

June 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年六月

一個基於蠕蟲的合作式防禦系統

研究生：李育松　　　　　　　　　　　　　　　　指導教授：曾憲雄博士

國立交通大學資訊科學系

摘要

隨著資訊科技不斷的演進，也伴隨著產生許多的變異物件。然而變異物件產生速度不斷的加快，專家尋找變異物件所花費的精神也越加繁重。VODKA 是一個發現變異物件的知識擷取方法，可以協助找出隱藏在真實世界中的變異物件。然而隨著變異物件產生速度不斷的加快，VODKA 提供的情境(Context)資訊太少，導致決策時需耗費較多的精神。因此，在本篇論文中，延伸之前的 VODKA，使它能提供更多資訊輔助領域專家分析變異物件。也就是說，在本地端有些確認程度(CF)較低的變異物件即使有情境資訊的輔助，在單一台機器上並不容易辨識，容易有不確認的情況。因此，提出一個合作式的變異物件分析專家系統，藉由多台 VODKA 的回報資訊，系統化的分析是否有變異物件產生。而在本篇論文的應用實例中，將延伸型的 VODKA 應用在電腦蠕蟲這個領域，結果顯示，實作此合作式的變異蠕蟲分析專家系統，分析多台的回報資訊，可以輔助領域專家發掘本地端不易確認的複雜變異蠕蟲。

關鍵字：變異物件、知識擷取、專家系統、電腦蠕蟲、惡意程式、變種蠕蟲

Collaborative Defending System for Computer Worms

Student: Yu-Sung Lee 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 growth of variant objects, domain experts might not be easy to keep up with the dramatically increasing knowledge. Although Variant Object Discovering Knowledge Acquisition (VODKA) is proposed to discover variant objects in our real world, it still provides insufficient context information and results in the heavy confirmation effort of domain experts. Hence, we propose extended VODKA to supply more context information for helping experts make correct decision in this thesis. However, several uncertain cases might not be discovered and learned in local environment because the context information might be not enough to determine whether it is a variant occurred in local or not. Therefore, a collaborative analysis expert system is proposed to solve those local uncertain cases according to the meta knowledge including environment factors and domain specific heuristic criteria. The construction of meta knowledge is also proposed based upon the Repertory Grid and Attributes Ordering Table to automatically generate corresponding collaborative analysis rules. Finally, the collaborative defending system for computer worms is implemented to evaluate extended VODKA. As a result, the implementation of collaborative defending system can assist domain experts to discover several sophisticated worms which can not be learned in the local environment.

Keyword : Variant Objects, Knowledge Acquisition, Expert System, Computer Worms, Variant Worms

誌謝

能順利的完成此篇論文，最首要感謝我的指導教授，曾憲雄博士，曾教授在我碩士班的兩年期間相當耐心的指導我的論文研究;從他身上學習到許多領導處事的技巧，寶貴的經驗讓我獲益匪淺，不甚感激。同時也感謝我的口試委員，黃國禎教授，蔡文能教授，和陳年興教授，他們給予的寶貴意見，讓我的論文研究更有價值。

接下來要感謝林順傑學長，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，接受我的詢問和探討，和協助我論文修改工作，深表感激。

此外必須感謝實驗室學長平日的諸多協助，特別是王慶堯學長及曲衍旭學長。同時也感謝實驗室同窗夥伴，黃柏智、吳政霖、邱成樑、陳瑞言、陳君翰、宋昱璋、林易虹等人在生活上和課業上的友情支持，深表感激。

最後我也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，也讓我有足夠的動力完成此篇論文，深表感激。

Table of Contents

摘要……………………………………………………………………………I
Abstract………………………………………………………………………II
誌謝……………………………………………………………………………IV
Table of Contents……………………………………………………………V
List of Figures………………………………………………………………VI
List of Tables………………………………………………………………VII
List of Algorithms…………………………………………………………VIII
List of Examples……………………………………………………………IX
Chapter 1. Introduction..1
Chapter 2. Related Work...3
2.1. Life Cycle of A Worm 3
2.2. Brief of Worms..4
2.3. The Difference Between Variant and Original Worms 6
2.4. Knowledge Acquisition Systems 9
2.5. Problems of VODKA...19
Chapter 3. Extended VODKA for Collaborative Environment.............................20
3.1. The Framework of Extended VODKA Environment 20
3.2. Extended VODKA 21
3.3. Collaborative Analysis Center 25
Chapter 4. Collaborative Defending System for Computer Worms. 33
4.1. The Framework of Extended VODKA for Variant Worm Discovering........34
4.2. Process in Local Extended VODKA...35
4.3. Process in Collaborative Analysis Center 42
4.4. Evaluation 50
Chapter 5. Conclusion and Future Work. 55
References 56

List of Figures

Figure 2.1：Worm Life Cycle……………………………………………….…………3
Figure 2.2：The Concept of VODKA Framework……..……………………………11
Figure 2.3：The Learning Flow of VODKA………..………………………………..13
Figure 2.4：The Decision Information to Experts…..………………………………18
Figure 3.1：The Framework of Extended VODKA Environment……………………20
Figure 3.2：Extended VODKA………………………………………………………22
Figure 3.3：Stage of Global Analysis Center………………………………………...25
Figure 4.1：The Collaborative Defending System for Computer Worms………...…33
Figure 4.2：Knowledge Decision Phase……..……………………………….…….36
Figure 4.3：System Profile of Local Host……………………..……………………..36
Figure 4.4：Infecting Path Table Acquisition……………………………………..…..38
Figure 4.5：Stage of Global Analysis Center for Computer Worms…………….…...42
Figure 4.6：Communication between Local and Collaborative Analysis Center..……43
Figure 4.7：Experiment Environment…….………………………………………..….51
Figure 4.8：The Variation of Confidence of Dataset Baseline……….………………52
Figure 4.9：The Variation of Confidence of Dataset Normal……….………………..53
Figure 4.10：The Variation of Confidence of Dataset Attack…………………...……53

List of Tables

Table 2.1 ： Infecting Path of CodeRed and Nimda…………………………………....7
Table 2.2：Repertory Grid of Three Nimda Family Members………………………...8
Table 2.3 ：Repertory Grid of Two CodeRed Family Members……….....…………...8
Table 2.4：The Repertory Grid of CodeRed Worm……………………..………....14
Table 2.5：The AOT of CodeRed Worm……..……………………………...………..14
Table 2.6：Partial Detection Rules Generated by EMCUD………………..………15
Table 2.7：Inference Log from Inference Engine……………………….……………15
Table 2.8 ：The New Variant Acquisition Table of CodeRed.II…………………….…..16
Table 3.1：Log Profile of Local Host…………………..…………………….………26
Table 3.2：Log Profile of Global Log Database……….……………………………..26
Table 3.3：System Profile of Worm Domain of Local Host…..……………………27
Table 4.1：Format of System Profile…………………….…………………………37
Table 4.2：System Profile of Local Host……………………………………………37
Table 4.3：Infecting Path of CodeRed………………………….……………………39
Table 4.4：System Profile of Local Host………………………………………………..39
Table 4.5：Infecting Path Table of CodeRed Family…………..…………….………39
Table 4.6 ：Inference Log from Inference Engine……………………………………40
Table 4.7：System Profile of Local Host for Yaha………………………..................41
Table 4.8 ：Infecting Path Table of Yaha Family………………………………...……..41
Table 4.9：Log Profile of Yaha………………………………………………….……43
Table 4.10 ：System Profile for Computer Worms…………………………………43
Table 4.11 ：The Repertory Grid of Yaha, Sobig and Gaobot…………………….…..45
Table 4.12 ：The AOT of Yaha, Sobig and Gaobot……….…………………………45
Table 4.13：Partial Detection Rules Generated by EMCUD…………..…….……46
Table 4.14：Mapping Table of Yaha, Sobig and Gaobot………....……….………46
Table 4.15：System Profile of Three Hosts………………………………….….……46
Table 4.16：The Acquisition Table of Gaobot……………………………..……50

List of Algorithms

Algorithm 2.1：The Algorithm of VODKA……………………………………………12

Algorithm 4.1：Infecting Path Table Construction Algorithm………………………38

List of Examples

Example 2.1：An Example of Learning Variant Worms…..…………………...……14
Example 4.1：Infecting Path Table of CodeRed……………………………………..38
Example 4.2：Mapping with System Profile and Infecting Path Table…………...…39
Example 4.3：Uncertain Case in Local VODKA………..……………………..……40
Example 4.4：An Example to Trace Collaborative Analysis Rules………………...45

Chapter 1. Introduction

As we know, knowledge based 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, such as disease diagnosis [6][13], investment prediction [2], or science [1]. Inference is the way that computer reasons according to the knowledge base which stores the domain expertise in the computer recognizable format. Embedded Meaning Capturing and Uncertainty Deciding (EMCUD) [7] was proposed to elicit the embedded meanings of knowledge and guide experts to decide the certainty degree of each embedded rule with embedded meaning for expending the coverage of generated rules. However, some embedded rules may be with low/marginally acceptable certainty factor (CF) values due to the weak suggestions of domain experts, and new variants derived from the well-known objects in many domains are incrementally developed due to the coming age of the knowledge explosion. Hence, Variant Objects Discovering Knowledge Acquisition (VODKA) [13] was proposed by Tseng et al, 2004 to collect sufficient information, which is the related ambiguous attributes due to the marginally acceptable CF values of original rules suggested by experts, for refining the original knowledge base to enhance the classification ability.

Although VODKA is a well knowledge acquisition method for helping experts clearly distinguish the new variants, a subset of the original object having some different characteristics, from original object, it has some problems such as providing insufficient context information resulting in the heavy confirmation effort of domain experts and might still lack of the ability for sharing the information collecting in autonomous area to notice the occurrence of variants. Hence, in the thesis we extend

VODKA including local and global enhancement, for helping domain experts discovering variant objects.

1) Local enhancement - Extending VODKA for supplying more context information.

In order to reduce the conformation effort of experts, we extend VODKA to provide more context information such as environment factors for helping experts make the correct decision.

2) Global enhancement – Extending VODKA for a collaborative analysis mechanism with expert system

Several uncertain cases might not be discovered and learned in local environment because the context information might be not enough to determine whether it is a variant occurred in local or not. Hence, a collaborative analysis framework is proposed to handle these uncertain cases. The collaborative analysis framework is a systematical analysis mechanism with expert system to discover variant objects according to the meta knowledge including environment factors and domain specific heuristic criteria.

Finally, we apply extended VODKA to discover variant worms for computer worm domain. The recent computer worms, which are self-propagating computer programs, are very difficult for experts to get and analyze the signatures because they have incredible sophisticated characteristics [8]. These Internet worms could, in a very short time, cause great damage to network and information infrastructure. In order to evaluate our proposed extended VODKA, the collaborative defending system for computer worms is implemented. As a result, the implementation of collaborative defending system can assist domain experts to discover several sophisticated worms which can not be learned in the local environment.

Chapter 2. Related Work

Before describing our research focus, life cycle of a worm, some brief descriptions of computer worms and the difference between original and variant worms will be first introduced. Furthermore, VODKA, a powerful method to learn the evidence of variants based upon the inference log of embedded rules, will be then introduced. Although VODKA is good for learning variants, it still has several weaknesses. Hence, an example will be finally introduced to point out some problems of VODKA.

2.1 Life Cycle of A Worm

A computer worm is defined as a process that can cause a copy of itself to execute on another machine. In [3], it anatomized the life cycle of computer worm including Initialization Phase, Target Scanning Phase, Attack Phase and Infection Phase which are shown in Figure 2.1.

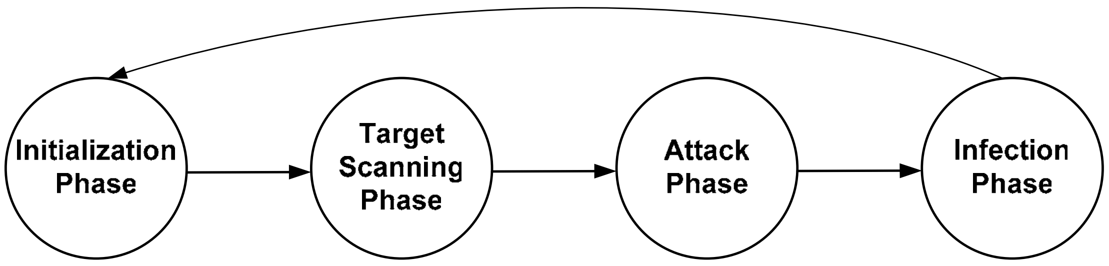

Figure 2.1 : Worm Life Cycle

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. In Target Scanning Phase, worms must discover a machine to infect by using predefined scanning techniques like sequential scanning, random scanning, pre-generated target lists, etc. [15]. Then a worm can actively spread it from machine to machine. After that, infected system may cause abnormal behavior to attack user host in Attack Phase and infect next victim in Infection Phase. The initialization, scanning, attacking and infection will continue cyclically.

2.2 Brief of Worms

Since worm is a self-replication program, it can be moved automatically from one host to another resulting in great damage. Therefore, the thesis is focusing on discovering variant worms to reduce the damage caused by worms. Several famous worms which will be used in our thesis are introduced.

In July 2001, the Code Red Worm [4] 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. Once installed, it began scanning for additional hosts to attack. Additionally, the worm used a Distributed Denial of Service Attack (DDoS) [12] against an IP of http://www.whitehouse.gov. Code Red used a random number generator to get new IP addresses to attack. The initial revision of Code Red hit 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.

In September 2001, the Nimda worm [5] was released on the internet. Again, Nimda can attack IIS servers with known vulnerabilities, but uses a different set of attack methods to do so. 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.

In February 2002, the Yaha Worm [11] was released on the Internet. The worm is a mass- mailing worm that uses its own SMTP engine to send itself to email addresses that exist in the Windows Address Book, the MSN Messenger contacts list, the Yahoo pager contacts list, the ICQ contacts list, and files that have extensions that contain the letters 'ht'. When Yaha is executed, it does the following: It sends itself to all the email addresses it finds in the infected system. It will modify registry key and attempt to send mail using information from the registry key. Also it will copy itself to the files, C:\Recycled\Msscra.exe and C:\Recycled\Msmdm.exe.

In October 2002, the Gaobot Worm [9] was released on the Internet. The worm also attempts to spread to all computers on the network, using a utility that connects to a remote computer on port 445. It copies the Woinggg.exe file across the network, and then executes it. It then connects to an IRC server and listens for commands. Upon execution, W32.Gaobot.Worm performs the following actions: It copies itself as %system%\Sysldr32.exe, modifies register key, connects to an IRC server on one of port 6667 and 9900, performs a Denial of Service attack on a specified server, open/close the CD-ROM drive and post the CD-Key for the some games to an IRC channel

In January 2003, the Sobig Worm [10] was released on the Internet. The worm sends itself to all the addresses it finds in the .txt, .eml, .html, .htm, .dbx, and .wab files. When the worm is executed, it does the following: It copies itself as %Windir%\Winmgm32.exe,.creates a %Windir%\Winmgm32.exe process, with the parameter of "start." and configures itself to start when you start Windows. The worm stores the addresses to which it sends the email messages in the file %Windir%\Sntmls.dat.

2.3 The Difference Between Variant and Original Worms

Based upon the worms described above, we can observe the difference between variants and original worms is quite minor. The variant worms can be generated by modifying the same malicious code which is easy to get in Internet. The differences can be compared in generation and diversity, which will be detailedly described in the followings. In the evolution of variant worms we can observe the relationship between original and variant worms.

2.3.1 Generation

When an unsafe configuration is discovered, hackers will start to generate new malicious code to attack. When an original worm is released by hackers, some anti-virus corporation like Symantec or Trend Micro will update their scanning engine by new virus patterns to avoid attacking. Domain experts will extract specific binary codes as the new virus patterns. These specific virus patterns are designed for this new original worm. In order to evade pattern matching by these anti-virus corporations, hackers usually modify the same source code of original worm. And then a new variant is generated. At this time these anti-virus corporations will retrain new virus pattern for variants to stop attacking and then hackers will generate another variants. The hackers and anti-virus corporations are competing with each other. These result in large amount variant worms in our real world.

2.3.2 Diversity

Because most variant worms are generated by slightly modifying the same original programs, they inherit the same primary attributes of original worms. So we can observe from technical documents that they usually have the same spreading ways and similar symptoms. Hence, the diversity between the great part of original and variant worms is in some ambiguous attribute value.

From this observation, domain experts will have this kind of prior knowledge to assist classify computer worms due to the similar spreading ways and symptoms. In Section 2.3.3 we introduce two real cases to verify our observation.

2.3.3 The Cases of Worm Family - CodeRed and Nimda

In order to clearly represent the diversity between variant and original worms, two real cases of famous worm family including CodeRed [4] and Nimda [5] are discussed in the followings. Since the technical documents of computer worms are usually unstructured, a formatted representation is needed for easily accumulating the knowledge of worms. According to the above description, two kinds of attributes including infecting path and symptoms are used to recognize the computer worms. We use Table 2.1 to show the infecting path of CodeRed and Nimda. They own their specific infecting path.

Table 2.1：Infecting Path of CodeRed and Nimda

Worm_Name	Infecting_Path

CodeRed	CVE-2001-0500(IndexServer2.0 and IIS6.0);
	CVE-2001-0506(IIS4.0 and IIS5.0)

Nimda	Email;CVE-2001-0154(MIME Header); CVE-2000-0884(IIS Web Server Folder Traversal);Data Share

Besides, the symptoms of Nimda family are listed as Table 2.2 with DoS_Type, New_Guest, Function_abnormal, Email_Attachment, TCP_Port, New_File and Open_Disk_Share, etc. Data type of attributes are Boolean, String, Integer or sets. According to Table 2.2, we can discover that Nimda family has the same primary attributes and might have different value of other attributes between the variant worms.

Table 2.2：Repertory Grid of Three Nimda Family Members

Object / Attribute	Nimda.A	Nimda.B	Nimda.E
DoS_Type	Email Flood	Email Flood	Email Flood
New_Guest	True	True	True
Function_Abnormal	Word、WordPad	Word、WordPad	Word、WordPad
Email_Attachment	readme.exe;readme.wav; readme.com	puta!!scr	Sample.exe
TCP_Port	25;80;137;138;139;445	25;80;137;138;139;445	25;80;137;138;139;445
New_Files	meXXXXX.tmp.exe	meXXXXX.tmp.exe	meXXXXX.tmp.exe; CSRSS.EXE; httpodbc.dll
Open_Disk_Share	True	True	True

In this case, Attribute(Email_Attachment, New_File) are the ambiguous value which could be changed with a new variants.

Table 2.3：Repertory Grid of Two CodeRed Family Members

Object / Attribute	CodeRed.A	CodeRed.B
DoS_Type	TCP Flood	TCP Flood
Threads	100	600
Backdoor	X	True
TCP_Port	80	80

Table 2.3 shows some major attributes of CodeRed family. Based upon the Table 2.3, we observe the CodeRed.B and CodeRed.A have similar symptoms. The difference between them is the number of automatically created threads and the creation of new attribute backdoor.

According to infecting path and symptoms, worms could be classified after analyzing the infected machines in laboratory. For example, if a new worm uses the same IIS Server Buffer Overflow, CVE-2001-0500 and CVE-2001-0506, and the major symptoms are the same as CodeRed Family, domain experts will classify this new worm into CodeRed family as a variant worm.

2.4 Knowledge Acquisition Systems

VODKA [14] is proposed to learn new variant objects from analyzing the acceptable embedded meaning of knowledge. The embedded knowledge is elicited by EMCUD [7] from the existing repertory grids. In order to understand these knowledge acquisition systems, firstly we describe knowledge acquisition and three kinds of knowledge acquisition approaches in Section 2.4.1. In Sections 2.4.2 and 2.4.3 two knowledge acquisition systems, EMCUD and VODKA, will be introduced.

2.4.1 Knowledge Acquisition

In order to obtain the knowledge of a special domain, knowledge acquisition is proposed to transfer the expertise of domain experts into knowledge bases. General speaking, there are three kinds of approaches for knowledge acquisition.

1. Interviewing experts by experienced knowledge engineers.

After interviewing the domain expert, knowledge engineers explicitly code the knowledge. But, it might be time consuming for the domain expert and knowledge engineers to understand each other.

2. Machine learning.

Given training cases, the machine learning approach will automatically learn some rules. But it still has two disadvantages. Firstly, there might be few or no available training cases in many application domains. Moreover, it is hard to understand the relation among cases.

3. Knowledge acquisition systems.

The knowledge acquisition system involves interviewing with the help of knowledge acquisition tools. It helps knowledge engineers work better in interviewing experts. Besides, deeper knowledge can be elicited using this approach such as repertory grid technique which gets domain experts to rank objects against concepts. The knowledge acquisition system solves the problem of communication between domain experts and knowledge engineers without the required training cases.

2.4.2 EMCUD (Embedded Meaning Capturing and Uncertainty Deciding)

EMCUD, a knowledge acquisition system, is proposed to elicit the embedded meanings of knowledge from the existing repertory grids. Additionally, it will also guide experts to decide the certainty degree of each rule with embedded meaning for extending 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 kinds of 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 the attribute dominates the object and an integer is represented for the relative important degree of the attribute to the object. The larger the integer is, the more important the attribute is to the object.

Using AOT, 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.

2.4.3 VODKA (Variant Objects Discovering Knowledge Acquisition)

Although EMCUD and other similar approaches could be rerun to acquire such knowledge from domain experts again to distinguish new variants from old objects, it might be costly and hard to obtain the knowledge due to the lack of sufficient information about variants. Therefore, the idea is to analyze the inference behaviors of weak embedded rules to construct the new variants acquisition table. In [14], a new iterative knowledge acquisition methodology, Variant Objects Discovering Knowledge Acquisition (VODKA), is proposed to provide the ability of grid evolution.

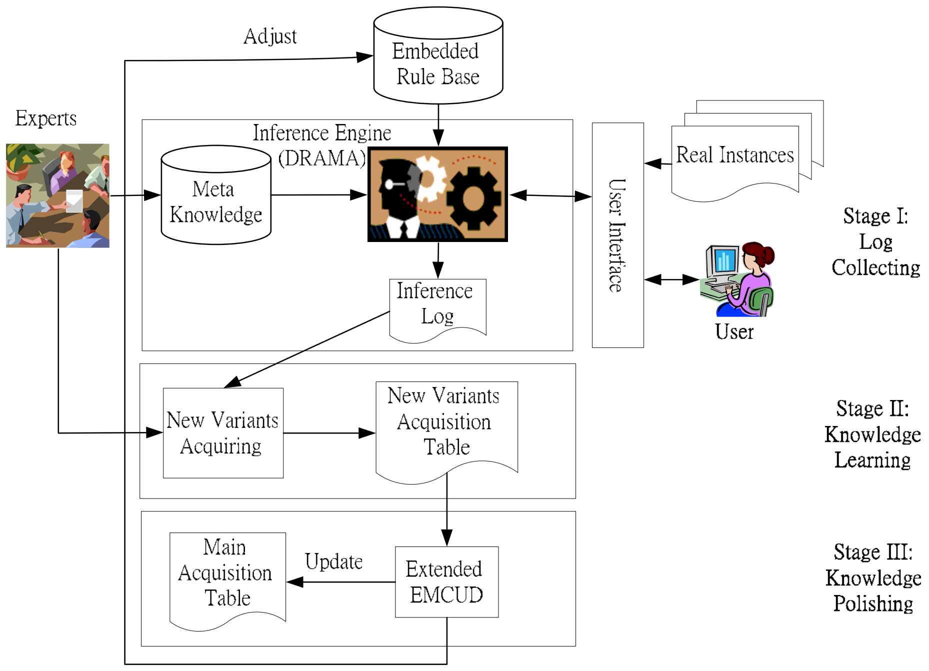

Figure 2.2：The Concept of VODKA Framework

Because the embedded rules with diverse CF values represent the different supports to classify objects, the ones with marginally acceptable CF might be triggered by some candidate of a new variant. In order to analyze the behaviors of these embedded rules, a VODKA framework is shown in Figure 2.2 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) 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, the new variants acquisition table based on the discovered knowledge could be generated by interacting with domain experts through the new variants acquiring procedure. Finally, the rules of new variants will be incrementally generated and the main acquisition table will be iteratively adjusted using E-EMCUD in Stage III. The algorithm of VODKA is shown as follows.

Algorithm 2.1: The Algorithm of VODKA

```
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.

2.4.4 The Learning Flow of VODKA

Figure 2.3 shows the learning flow of VODKA. We periodically analyze the inference log to discover variant objects. If the CF is higher, we can determine a new object immediately. If the CF is an acceptable value, find frequent itemsets and ask experts if these are variant objects or not. After decision process of domain experts, they will tell us variants or not. And VODKA provides some operations to let experts quickly generate new variant acquisition table. Finally, use EMCUD to generate original and embedded rules to enhance our embedded rule base.

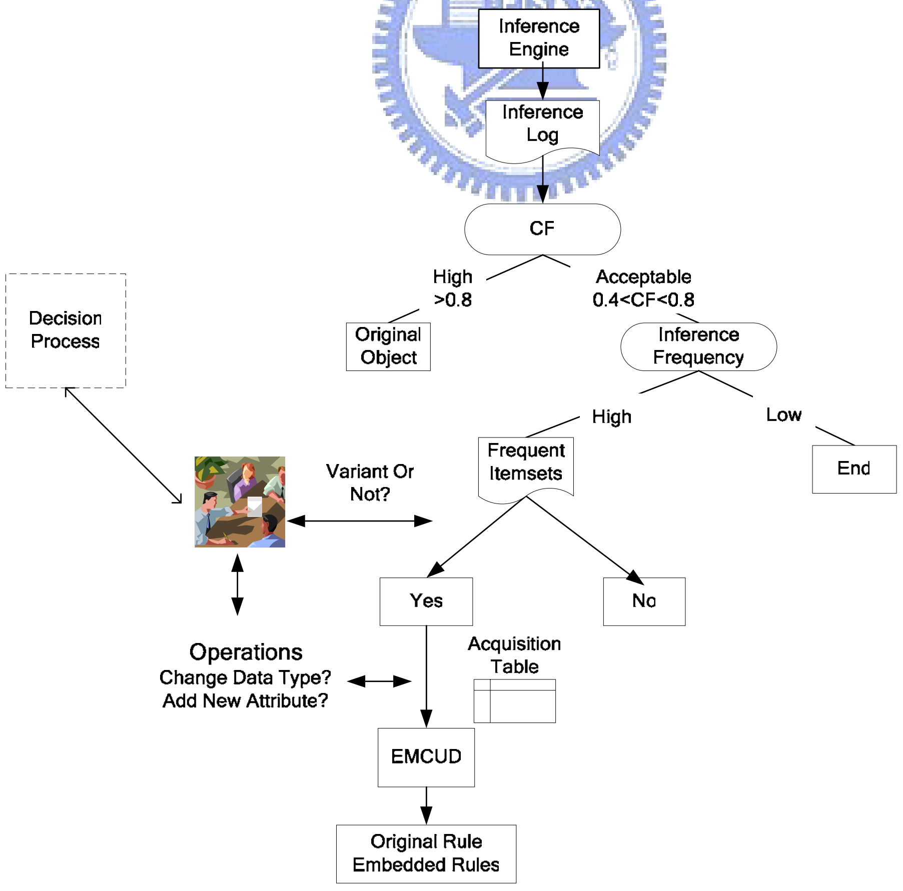

Figure 2.3：The Learning Flow of VODKA

2.4.5 An Example of Learning Variant Worms

In Example 2.1, we apply VODKA to learn variant worms. By this example, we introduce the learning mechanism of VODKA and then indicate the weakness of VODKA.

Example 2.1：An Example of Learning Variant Worms

The Repertory Grid is used to acquire original worm and an AOT represents the relative importance to attributes. After generating repertory grid and AOT, we can use EMCUD to generate original and embedded rules listed in Table 2.6.

Table 2.4: The Repertory Grid of CodeRed Worm

Object / Attribute	CodeRed (O_1)
100-Thread (A_1)	True
System_Reboot (A_2)	X
DoS_Type (A_3)	TCP Flood
Email_Attachment(A_4)	X
Antivirus_Firewall_Abnormal(A_5)	X
TCP_Port (A_6)	{80}
New_File(A_7)	X

Table 2.5: The AOT of CodeRed Worm

Object / Attribute	CodeRed (O_1)
A_1	2
A_2	X
A_3	1
A_4	X
A_5	X
A_6	3
A_7	X

Table 2.6: Partial Detection Rules Generated by EMCUD

Rule #	IF Part							Then Part	CF
	A_1	A_2	A_3	A_4	A_5	A_6	A_7	Object	
$R_{1,0}$	True	-	TCP flood	-	-	{80}	-	CodeRed	0.7
$R_{1,1}$	True	-	¬TCP flood	-	-	{80}	-	CodeRed	0.5
$R_{1,2}$	False	-	TCP flood	-	-	{80}	-	CodeRed	0.4

In this example, assume the fired sequence of some embedded rules of CodeRed worms are given as follows.

Table 2.7：Inference Log from Inference Engine

Rule #	A_1	A_2	A_3	A_4	A_5	A_6	A_7	Object	CF
$R_{1,2}$	600	-	TCP flood	-	-	{80}	-	CodeRed	0.4
$R_{1,2}$	600	-	TCP flood	-	-	{80}	-	CodeRed	0.4
$R_{1,1}$	100	-	-	-	-	{80}	-	CodeRed	0.5
$R_{1,2}$	600	-	TCP flood	-	-	{80}	-	CodeRed	0.4
$R_{1,2}$	150	-	TCP flood	-	-	{80}	-	CodeRed	0.4
$R_{1,1}$	100	-	-	-	-	{80}	-	CodeRed	0.5
$R_{1,2}$	600	-	TCP flood	-	-	{80}	-	CodeRed	0.4
$R_{1,2}$	600	-	TCP flood	-	-	{80}	-	CodeRed	0.4
$R_{1,2}$	600	-	TCP flood	-	-	{80}	-	CodeRed	0.4
$R_{1,2}$	300	-	TCP flood	-	-	{80}	-	CodeRed	0.4

Assume the minimal support is set to 30%, the frequent itemsets will be obtained. For example, if a the frequent itemset L1=(A_1=600) is satisfied, the VODKA will ask the expert to confirm such itemset if it belongs to certain variant.

VODKA will ask the following questions

VODKA：Does the attribute-value pair (A_1=600) belong to any new variant object?

/* Decision Process of experts*/

Expert：Yes

VODKA：What is the name of the new variant object?

Expert：CodeRedII

VODKA：Is the data type of A_1 required to be changed?

Expert：YES

VODKA：What data type do you want to change? 1.Boolean 2.Integer 3.Float 4.String 5.Set of values 6. Range of values

Expert：2

VODKA：Is any new attribute required to be added?

Expert：No

VODKA：Can the Single Value data type be used to change the original Boolean data type of A_1?

Expert：YES

VODKA：What is the new name and new value set of the attribute A_1?

Expert：Threads, (100, infinite).

Therefore, the new variant acquisition table of CodeRed.II shown in Table 2.8 will be generated.

Table 2.8：The New Variant Acquisition Table of CodeRed.II

Object / Attribute	CodeRed	CodeRedII
Threads (A_1)	100	600
System_Reboot (A_2)	X	X
DoS_Type (A_3)	TCP flood	TCP flood
Email_Attachment(A_4)	X	X
Antivirus_Firewall_Abnormal(A_5)	X	X
TCP_Port (A_6)	{80}	{80}
New_File (A_7)	X	X

Consequently, an original rule and an embedded rule will be generated by EMCUD.

IF (Threads=600) AND (DoS_Type=TCP flood) AND (TCP={80}) THEN CodeRed.II CF=0.8

IF ¬(Threads=600) AND (DoS_Type=TCP flood) AND (TCP={80}) Then CodeRed.II CF=0.4

2.4.6 The Inefficiency of VODKA

Although VODKA could help expert identify variants derived from original worms, the first step, "VODKA asks does the certain attribute-value pair belong to any new variant object?" will cause much effort of expert. For example, domain experts need to consult more information to make a decision due to the simple questions asked by VODKA. But VODKA provides only the inference log of frequently fired embedded rules which includes less information to assist expert. Hence, we anatomize the process effort of domain experts. Firstly, they must know the environment setting of local system. Besides, the symptoms of the host are also important to experts decide whether the attack happens or not. However, these two decision information, environment factors and symptoms, could be used to assist experts make the suitable decision.

After considering above information, we can get the following decision information.

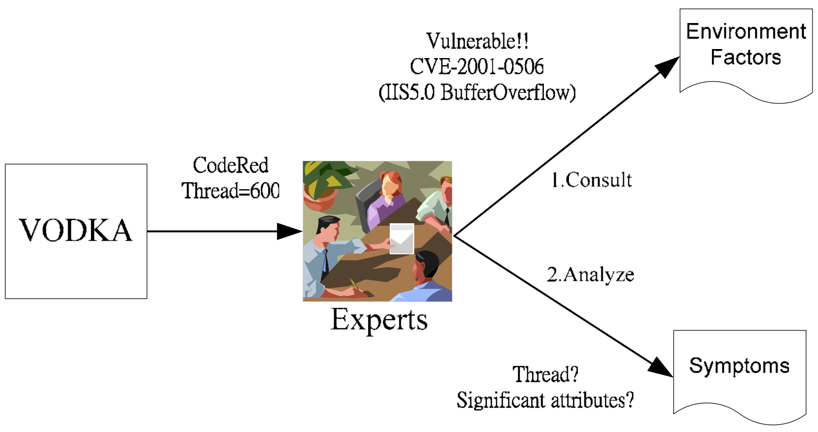

Figure 2.4：The Decision Information to Experts

Firstly, we can consult system profile (environment factors) of local host to identify if any possible vulnerabilities could be exploited to infect our system, which is called infecting path. If the infecting path is available, it might have high possibility to be infected by some malicious worms. Therefore the following message could be obtained for recording such information.

No patch IIS5.0 BufferOverflow (CVE-2001-0506)

=>CodeRed worm family has the way to exploit

=>increase CF

Secondly, the detected attributes will be examined whether attribute is significant or not to decide the degree of the recognition of worms.

The embedded rule of $R_{1,2}$ is fired when conditions partially match A_3 and A_5. Now we want to know the attribute-value pair (Thread=600) is a significant attribute or not.

=>the attribute (symptom) is like a CodeRed worm

=> increase CF

Experts might have lower confidence for some embedded rules $R_{1,2}$ for recognizing worms. However, they might incrementally enhance the confidence according to above information provided and concluded the variant is recognized. However, not all of such case could be stronger to make a decision immediately, several cases might happen fewer and confuse experts to make a correct decision.

This is called uncertain case.

2.5 Problems of VODKA

As mentioned above, two issues are concluded to point out the weakness of VODKA.

1. Information insufficiency

Some information can be collected ahead, but VODKA provides insufficient information. This increases domain experts' effort when determining variant objects.

2. Hard to make a decision

Some cases are hard to make a decision due to insufficient information. Our collected embedded knowledge may be disturbed by some legitimate software like mass email software, P2P , etc. During this situation the answer may be uncertain due to the weakness of embedded knowledge or information insufficiency.

So, the thesis mainly focused on solving the above problems. Therefore, we propose a methodology to extend VODKA including local and global enhancement.

1. Local enhancement - Extending VODKA for supplying more context information

With the rapid growth of variant objects, insufficient context information will increase confirmation effort of experts. Therefore, we enhance VODKA to provide more context information to reduce confirmation effort of experts.

2. Global enhancement - Extending VODKA for a collaborative analysis mechanism

We propose a collaborative analysis mechanism to solve local uncertain cases when discovering variant objects. By the collaborative analysis framework, VODKAs can have the ability to share information. And then, domain experts can make decisions with global views instead of local views.

Chapter 3. Extended VODKA for Collaborative Environment

Since VODKA [14] has been proposed to learn the variants from the original objects according to the inference log of embedded rules, it still has some problems such as providing insufficient context information resulting in the heavy effort of domain experts. However, some information in each domain could be prepared in advance for assisting expert to easily recognize the new suggested variants. Moreover, several uncertain cases might not be discovered in local environment because of insufficient context information. Before, VODKA is lack of collaborative ability to share local collected information of uncertain cases to learn more variants in different environment. Therefore, the extended VODKA for collaborative environment is proposed to collect more and more information from multiple VODKAs.

3.1 The Framework of Extended VODKA Environment

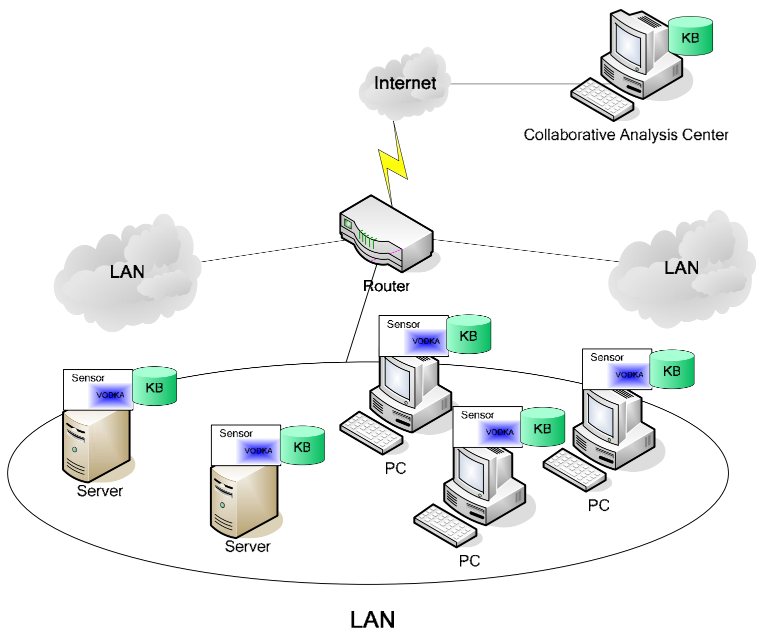

Figure 3.1：The Framework of Extended VODKA Environment

Figure 3.1 shows the framework of extended VODKA environment, where each local sensor deployed with VODKA learning module is to discover variant objects with adopting Knowledge-Base. However, some embedded rules with low CF might be not enough to conclude any results in local VODKA even if it consults more context information including the environment variables, etc. However, some variant objects may appear anywhere in network environment regardless of embedded rules with high CF or low CF. As we know, the embedded knowledge of higher CF can be discovered in local but the lower CF can not. By collecting and analyzing information of all uncertain cases periodically in collaborative analysis center, the information from multiple VODKAs could help experts make a decision easily with global views.

3.2 Extended VODKA

In order to enhance VODKA, we consider environment factors of domain to help experts making correct decision. **Firstly**, we supply the environment factors as context information for experts. **Secondly**, if the collected information is not enough to make decision by experts, such weak embedded knowledge will be reported to collaborative analysis center for further analysis.

To clearly describe our extended VODKA, Figure 3.2 shows the extended VODKA including two parts (the Environment Factors, the Collaborative Analysis Center) to help experts recognize the new variants.

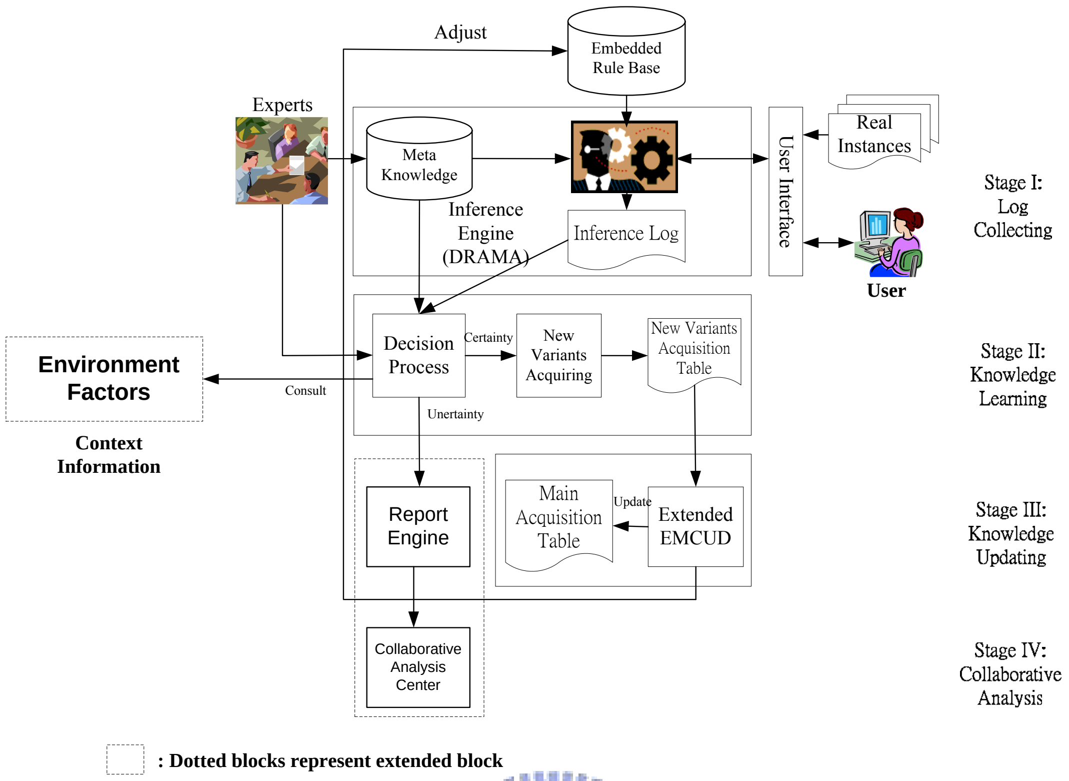

Figure 3.2 : Extended VODKA

The whole framework extends three stages to four stages including Log Collecting Stage, Knowledge Learning Stage, Knowledge Updating Stage, Collaborative Analysis Stage, as shown in the dotted blocks, where represent the extended components. Each stage is described as follows.

Stage I : Log Collecting

In Log Collecting Stage, the related inference log about embedded rules with marginal acceptable CF is collected. And the inference log which is collected from inference engine will be learned in knowledge learning Stage.

Stage II : Knowledge Learning

In knowledge learning stage, the candidate knowledge from inference log will be learned using data miming technology. In addition to ask experts to decide which new variant objects the candidate knowledge belongs to, the extended framework will consider more context information to reduce confirmation efforts of experts. If the context information is enough to make a decision, a new variant will be confirmed and the knowledge base will be updated in Knowledge Update Stage. Otherwise, experts will say "uncertainty" and then we will report to Collaborative Analysis Center for further analysis.

Stage III：Knowledge Updating

If the variants are generated by Knowledge Learning Stage, the new original and embedded rules will be obtained using extended EMCUD with the new acquired repertory grid and corresponding AOT. And then the embedded rule base will be updated to refresh the original knowledge.

Stage IV：Collaborative Analysis

The Collaborative Analysis Center will explain the uncertain knowledge learned by local VODKA. In this Stage, more information of multiple local VODKAs could be obtained to collect more evidence of variants. Hence, the decision confidence can be accumulated based upon the meta knowledge including environment factors and domain specific heuristic criteria.

3.2.1 Environment Factors

Because each variant object has its own growing environment, VODKA does not consider the environment factors. We give a verification of the environment factors, which can be used to help expert make a decision. For example, the computer worm domain, which is a self-propagating program via a network to threat the Internet very much, is one of the domains with dynamic environment configurations. With the evolution, large amount of variant worms are generated by hackers. And each worm has its own growing systems or applications(environments). The extended VODKA provides such information to help deciding variant objects. When making a decision, environment factors will be the context information for experts to reduce their confirmation effort of searching such information. The environment factor may be represented in many kinds of explanation to fit different domain. It could be a system configuration, network configurations, etc.

We give an example of CodeRed worm family which uses the buffer Overflow of IIS Server (CVE-2001-0500; CVE-2001-0506). As a result, the IIS Servers with vulnerability of CVE-2001-0500 and CVE-2001-0506 are the growing environment of CodeRed worms. In other words, CodeRed family will not grow without these conditions. So the vulnerable environment of IIS Servers is context information for experts to decide variant objects of CodeRed.

3.2.2 Collaborative Analysis Center

With the additional context information, some variants might be too sophisticated to discover and learn in local environment because the context information might be not enough to determine whether it is a variant occurred in local or not. Therefore, more information should be collected to overcome such problem to learn the variants. Hence, such un-decidable inference log will be transferred to collaborative analysis center for analyzing and learning since the variants may occur anywhere. Finally, the collaborative analysis center could learn and make more correct decisions with global views according to meta knowledge including environment factors and domain specific heuristic criteria. More details about collaborative analysis center will be represented in following section.

3.3 Collaborative Analysis Center

Since local sensors might not have enough information to learn the sophisticated variants, the collaborative analysis center collected all un-decidable inference log to learn and discover such variants with global views.

The collaborative center consists of three stages including *Evidence Collecting Stage*, *Variants Analysis Stage*, *Knowledge Updating Stage*.

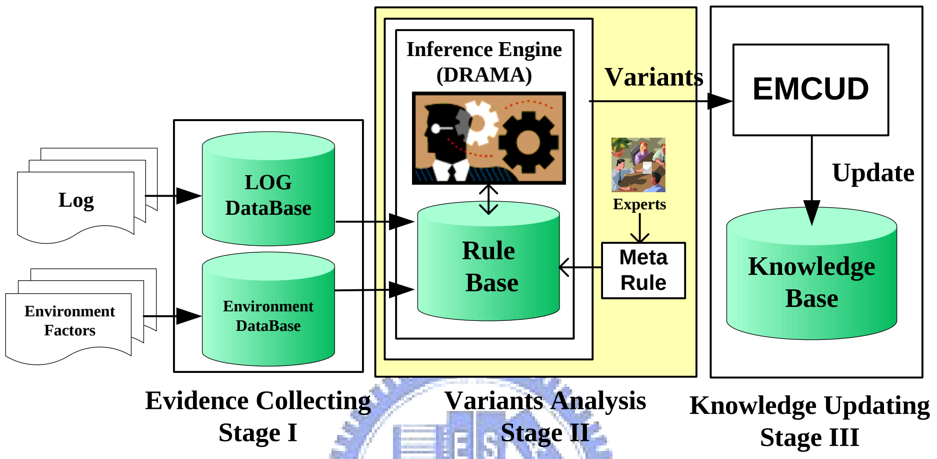

Figure 3.3：Stage of Global Analysis Center

3.3.1 Evidence Collecting Stage

In Evidence Collecting Stage, the inference Log periodically collected from local sensors and the Environment Factors of each sensor registered initially are collected and stored into database. Inference Log is designed to transfer during a regular period of time, which the period could be adjusted dynamically, but environment factors would be updated when changing from local sensors.

The format of inference log profile and environment factors are described as follows.

1. Log Profile

In order to collect uncertain knowledge from local hosts and store it in LOG Database, the log format, consisting of rule name, collecting attribute values and CF of rules, of local sensor is designed.

Table 3.1：Log Profile of Local Host

$R_{i,j}$	Att_1	Att_2	…	Att_k	$CF(R_{i,j})$

When the inference log is reported from Local VODKA to Collaborative center, the sensor ID of local host and the reporting time will be appended to these records. The format is listed below where the Time field record the reporting time ($T_1,T_2,T_3,\ldots$) from local sensors, ID field records the IP address of local hosts, $R_{i,j}$ means the j-1 embedded rule of object i and $R_{i,0}$ is the original rule of object i, Att_k is the value of k^{th} attribute, CF records the Certainty Factor of each rule.

Table 3.2：Log Profile of Global Log Database

Time	ID	$R_{i,j}$	Att_1	Att_2	…	Att_k	$CF(R_{i,j})$

2. Environment Factors

For different domain, experts can define their own environment factors they want to refer.

For example, in the worm domain world, worms will exploit the wrong configuration of victim system to infect. Hence, the environment factor is system profile from local sensors including the IP addresses, the Operation system version, the provided services, the patching level, etc. Hence the local system profile will be collected as our environment factor for worm domain. If the system exists some vulnerabilities with some service, it might have the risk to be infected by corresponding worms. The information of system profile shown as follows will be consulted to help expert making decision in global center.

Table 3.3：System Profile of Worm Domain of Local Host

ID	OS	(Application <-> Vulnerability)
140.113.*.*	Windows Xp	Internet Explorer5.5 <-> CVE-2001-0154(MIME header)

ID：140.113.*.*：the local IP address

OS：the operation system

(Application <-> Vulnerability)= Internet Explorer5.5 <-> CVE-2001-0154

Internet Explorer5.5 with MIME header vulnerability

CVE-2001-0154 is the standard vulnerability name defined by CVE.

So, collaborative analysis center will receive log and related environment factors as decision information and keep them in global database.

3.3.2 Variants Analysis Stage

With the technology revolution, new variant objects can be generated rapidly than before. If we do not have a systematical and automatic analysis mechanism, we must be busy in discovering new variants. Hence, the expert system technology is used to solve these problems.

In the construction of rule base, a confidence value from experts' views is used to be accumulated. During the inference process, if the accumulated confidence of certain object meets a predefined confidence threshold. A variant object is discovered by our system.

About the construction of rule base, we will describe it detailedly in Section 3.3.4.

3.3.3 Knowledge Updating Stage

After analyzing the inference log, several variants could be discovered. When a new variant is determined, the original and embedded rules of the new variant will be generated using EMCUD. Therefore, the defending knowledge about new variant will be updated into the original knowledge base.

In order to well-maintain the collaborative analysis center, the meta knowledge should be constructed in advance and the systematical process to construct such meta rules is needed.

3.3.4 Meta Knowledge Constructing

For different domain, different expertise should be extracted to handle multiple reports from local sensor with VODKA module. Hence, domain experts can define their domain criterions to construct meta rules, named collaborative analysis rules, in variant analysis stage. Moreover, each rule would be appended with a value named confidence value. It means the accumulated confidence of experts when corresponding condition of variants is matched. If it is defined higher, large number of alerts might cause experts much effort. If it is defined lower, the system would generate slowly or miss variants. Hence, it must be well defined. If the reporting inference log satisfies certain collaborative analysis rules, the confidence value of each object will be increased to record the behaviors of such report. Hence, the confidence value of each object can be easily accumulated to monitor the distributed behavior of variants when corresponding rules are fired. When confidence value of each object exceeds a predefined confirmable threshold, it possibly represents a variant appeared.

Besides, different domain can increase their criterions. Now we will introduce some basic criterions to construct collaborative analysis rules listed as follows.

Criterion 1：Accumulate confidence according to the importance of attributes

Every report of candidate variant object has different importance. The importance can be seen in original AOT defined by experts. Hence, every report will accumulate our degree of confidence and the conjunction of attributes has different confidence value. If candidate variant object has more important attributes to this object class, we have higher confidence about this report.

To generate collaborative analysis rules, the information of AOT, which records the relative importance of each attribute to each object, will be reused to define the important degree of attributes. The attributes can be mapped into three parts including major attributes, secondary attributes and minor attributes. However, minor attributes which mean less important attributes to this object class can be ignored. Major attributes mean the important attributes for recognizing the object class and secondary attributes mean they could be negated to capture embedded meanings. If the events satisfying whole major attributes and secondary attributes, confidence obtained is higher than only satisfying major attributes.

For example, assume the range of AOT values is from 1 to 5, D and X. According to the importance of attributes, we map {D, 5, 4} to major attributes M, {3, 2} to secondary attributes Sand {1, X} to minor attributes. And then a confidence value of collaborative analysis rule will be normalized by the following formula.

$$Conf_{object} = Conf_{low} + \frac{(Conf_{high} - Conf_{low})}{(W_1 + W_2 + ... + W_k)} * (SUM(W_j))$$

Notation:

$Conf_{high}$: the highest accumulation of confidence.

$Conf_{low}$: the lowest accumulation of confidence.

$Conf_{object}$: the confidence of this collaborative analysis rule of object.

If the collaborative analysis rules only match major attributes, the Conf$_{object}$ is equal to Conf$_{low}$. If the collaborative analysis rules match major and secondary attributes, the Conf$_{object}$ is equal to Con$_{high}$. If the collaborative analysis rules match major and partial secondary attributes, the Conf$_{object}$ is equal to formula.

Hence, the Criterion1 means we accumulate more confidence when the reports match more importance attributes to this object class.

According to these methods we can generate collaborative rules defined to different objects and the general form of meta rule matched criterion1 is shown as follows.

MR1: **IF** (M_fully) **AND** (S_fully)

Then increase Conf$_{object}$ by Conf$_{high}$ **AND** set flag$_{object}$ = 1

MR2: **IF** (flag$_{object}$ =0) **AND** (M_fully) **AND** (S_partial)

Then increase $Conf_{object} = Conf_{low} + \frac{(Conf_{high} - Conf_{low})}{(W_1 + W_2 + ... + W_k)} * (SUM(W_j))$ **AND**

set flag$_{object}$ = 1

MR3: **IF** (flag$_{object}$ =0) **AND** (M_fully)

Then increase Conf$_{object}$ by Conf$_{low}$ **AND** reset flag$_{object}$ = 0

Notation：

M_fully：major attributes fully match

S_fully：secondary attributes fully match

S_partial：secondary attributes partially match

Conf$_{object}$：the confidence value of each object is initially set to 0

Conf$_{high}$：the highest accumulation of confidence

Conf$_{low}$：the lowest accumulation of confidence

$Flag_{object}$：A flag to each object is initialized to 0. Its goal is to infer the first matching rule.

$W_1+W_2+\ldots+W_k$：the sum of all AOT value of secondary attributes

$SUM(W_j)$：the sum of AOT value of partially matching secondary attributes

Criterion 2：Consider environment factors

Each variant object will have its own environment factors. If the system environment is matched, the confidence to make a decision will also increase.

Notation：

Object.ID.SystemProfile = Environment(Object)：It means that if system (ID.SystemProfile) has the growing environment for this object, our confidence will increase $Conf_{SystemProfile}$

```
MR4: IF Object.ID.SystemProfile = Environment(Object)
    Then increase $Conf_{object}$ by $Conf_{SystemProfile}$
```

Criterion 3：A predefined threshold of discovering variant objects

If the accumulated confidence of certain object exceeds the threshold from expert views, a variant object is discovered from our system.

VOthreshold：the threshold of discovering variant objects

```
MR5: IF $Conf_{object}$ > VOthreshold
    Then discover variant object
```

Criterion 4：A timeout period to reduce confidence

A Half-life condition is used to reduce false alarm since some adventitious cases are observed. According to above criterions, the confidence value will increase unlimitedly. However, if we can not confirm a variant object after a long period of time, we will decrease our confidence because it may be considered as a noise.

The criterion is listed as follows.

MR6: **IF** $Time_{period}=TH_{period}$

Then decrease CF_{object} to half

Notation：

$Time_{period}$：the time from previous reducing time

TH_{period}：A predefined time interval to reduce confidence

In this chapter, we propose a methodology of using Extended VODKA to learn variant objects more efficiently. Moreover, VODKA is a learning tool which can be appended to any knowledge-base system. In order to verify our methodology we apply VODKA to computer worm domain as our example in next Chapter.

Chapter 4. Collaborative Defending System for Computer Worms

The easy access and wide usage of the Internet make it more convenient for technical research and information exchange. Most of intrusion behaviors become more complicated because of combining several signatures of previous intrusions. The recent computer worms, which are self-propagating computer programs across a network exploiting security or policy flaws in widely-used services, are very difficult for experts to get and analyze the signatures because they have incredible sophisticated characteristics [8]. With the evolution of original worm, large amount of variant worms are continually generated to infect our computers. Because Extended VODKA can help us to discover these variant worms, we apply it to this domain in the chapter.

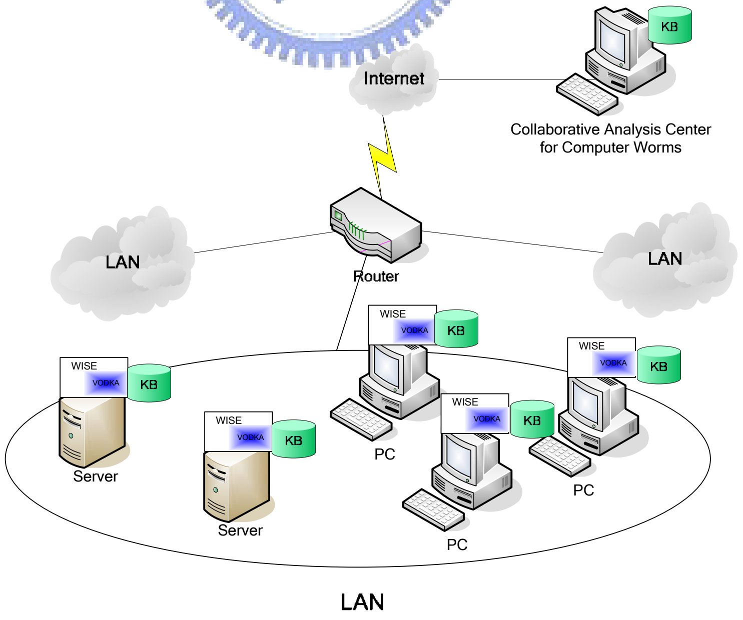

Figure 4.1：The Collaborative Defending System for Computer Worms

As mentioned above, the collaborative defending system for computer worms shown in Figure 4.1 is proposed to learn more variants from multiple sensors with

VODKA. In the computer worm domain, the sensor is named WISE, which is a worm immune service expert system. WISE is to help us diagnose our system far away from worm attacking. It uses the VODKA to be a learning module to increase knowledge of variant worms. Besides, a collaborative analysis center for computer worms is used to analyze the uncertain cases of variants in local.

Section 4.1 introduces the framework of Extended VODKA for variant worms discovering. Section 4.2 introduces the learning process in local VODKA. Section 4.3 introduces the collaborative analysis center to analyze the uncertainty of local VODKA. Section 4.4 evaluates our collaborative defending system for computer worms.

4.1 The Framework of Extended VODKA for Variant Worm Discovering

As we know, worms will infect wrong configured system. And each worm family has different infecting way to infect. If the system is vulnerable, worms will have environments to grow. Therefore, the system configuration, system profile is referred to as the environment factors. In computer worm domain, the environment factors include the infecting path and system profile will be considered.

However, some behaviors of worms look like normal behaviors such as mailing softwares, P2P , etc.. It might result in the uncertain decision even if we consult additional infecting path and system profile, which can be collected and used to analyze and solve the local uncertainty. Hence, if the uncertainty continually exists, we will deal with these problems with global views in Section 4.3.

4.2 Process in Local Extended VODKA

Extended VODKA considers context information. Therefore, the context information such as infecting path analysis, are proposed to reduce the confirmation effort of domain experts. The infecting path analysis is the environment factors for computer worm domain.

Now we will introduce them as follows.

4.2.1 Knowledge Decision Phase

The environment factors in StageII described in Figure 3.2 in computer worm domain can be treated as infecting path analysis. Because most variant worms use the same vulnerabilities to exploit, the system can be determined to have vulnerabilities by mapping Infecting Path Table and System Profile. Such context information will be consulted to make a decision.

Figure 4.2 shows that if we capture candidate embedded knowledge from VODKA. Before confirming by experts, we will analyze first and supply context information for suggesting experts. The infecting path analysis will tell us vulnerable or not by mapping infecting path of original worms and system profile of local host. With the context information, we have two conditions- certainty or uncertainty. Certainty means the evidence is powerful enough to confirm. Uncertainty means the evidence is not enough.

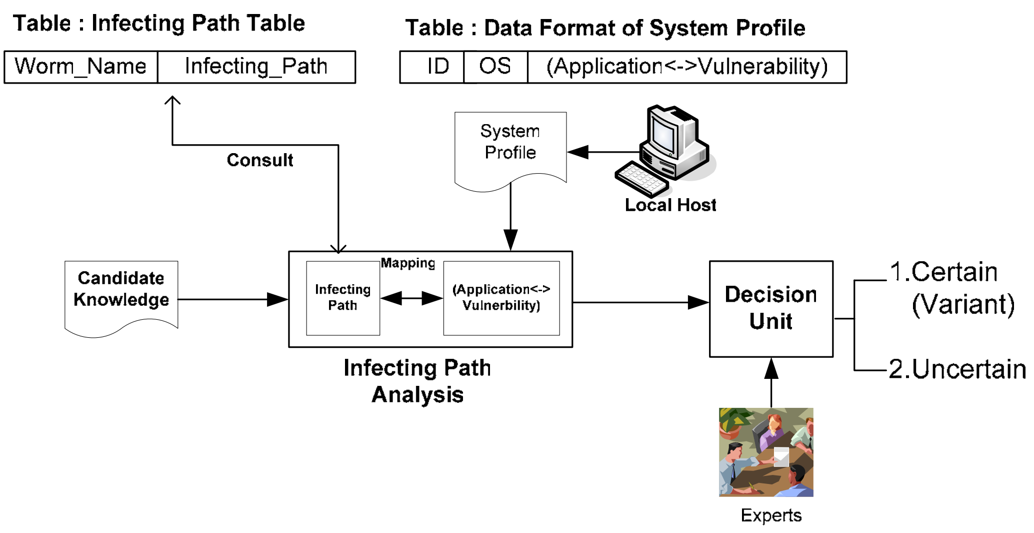

Figure 4.2 : Knowledge Decision Phase

The generation of the context information which will decrease the weakness of VODKA are described as follows.

4.2.1.1 Infecting Path Analysis - System Profile and Infecting Path Table

A. System Profile

Some worms are designed to infect wrong configured system and some worms exploit vulnerabilities of system or applications. If we have not patched our system frequently, it is easy to be infected by specific worms based upon the vulnerability of local host recorded in system profile.

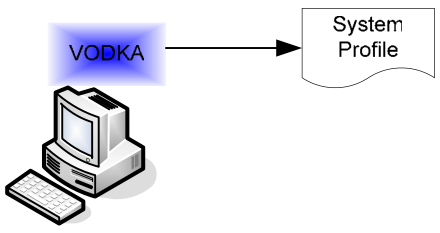

Figure 4.3 : System Profile of Local Host

How to get System Profile ?

When installing VODKA, the local system profile including ID, OS version and vulnerable applications can be recorded. Once the system profile is modified, VODKA will update it at the same time.

Now, the system profile is defined according to the worm domain.

Data format of System Profile

Definition：

Table 4.1：Format of System Profile

ID	OS	(Application <-> Vulnerability)
140.113.*.*	Windows Xp	Internet Explorer5.5 <-> CVE-2001-0154(MIME header)

Notation：

ID：Host IP

OS：Operation System

(Application <-> Vulnerability)：This represents which application is vulnerable

Data Type

ID：IPv4

OS：String

(Application <-> Vulnerability)：String

The following is a simple example to show our system profile

Table 4.2：System Profile of Local Host

ID	OS	(Application <-> Vulnerability)
140.113.*.*	Windows Xp	InternetExplorer5.5 <-> CVE-2001-0154(MIME header)

ID：140.113.*.*：the local IP address

OS：The operation system is Windows XP

(Application<->Vulnerability)=(InternetExplorer5.5<->CVE-2001-0154)

means

InternetExplorer5.5 with MIME header vulnerability

CVE-2001-0154 is the standard name defined by CVE

B. Infecting Path table

Because most variant worms use the same vulnerabilities to infect the system, the infecting path table to acquire vulnerabilities of original worms is proposed.

Infecting Path table

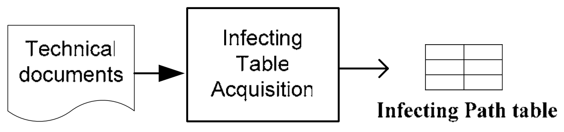

Figure 4.4：Infecting Path Table Acquisition

The infecting path table will be generated in Log collecting Stage. It accompanies with the generation of original rules (original worm). A simple knowledge acquisition is also proposed to acquire the infecting path of original worms.

The infecting table construction algorithm is shown as follows:

Algorithm 4.1: Infecting Path Table Construction Algorithm

```
Input: The worm domain know-how and skeletal of worm
Output: The Infecting Path of worm
Step1: List all elementary knowledge objects according to technical documents.
Step2: Transfer each infecting path into the Infecting Path Table.
```

The following example is given to construct infecting path table of CodeRed.

Example 4.1：Infecting Path Table of CodeRed

In this example, we construct infecting path table of CodeRed. Domain experts will extract infecting path of each original worm from technical documents. Therefore, the CVE-2001-0500, which is the Indexing Service2.0 and IIS6.0 vulnerability, and the CVE-2001-0506, IIS4.0 and IIS5.0 vulnerability, are exploited by CodeRed will be extracted in Step1. Then in Step2, the knowledge objects will be stored into

Infecting Path Table, shown in Table 4.1, with the pair of original name of Worm and the Infecting Paths exploited by the worm, where the data type are both string. Hence, we can obtain the following infecting path table of CodeRed after the process.

Table 4.3：Infecting Path of CodeRed

Worm_Name	Infecting_Path
CodeRed	CVE-2001-0500（Indexing Service2.0 and IIS6.0） CVE-2001-0506（IIS4.0 and IIS5.0）

C. Mapping with System Profile and Infecting Path Table

If local VODKA learns a new variant worm, it may map with system profile and infecting path table of this new variant worm. If the system is vulnerable, the decision confidence will be increased.

The following example is given to illustrate how to map with system profile and infecting path table.

Example 4.2：Mapping with System Profile and Infecting Path Table

Hence, VODKA now can suggest expert that a new variant worm derived from original CodeRed is learned. By mapping with system profile and infecting path table, we can know the system is vulnerable for CodeRed.

Table 4.4：System Profile of Local Host

ID	OS	(Application <-> Vulnerability)
140.113.87.175	WinXp	IIS4.0 <-> CVE-2001-0506

Mapping

Table 4.5：Infecting Path Table of CodeRed Family

Worm_name	Infecting Path
CodeRed	CVE-2001-0500（Indexing Service2.0 and IIS6.0） CVE-2001-0506（IIS4.0 and IIS5.0）

Hence, we can know the IIS4.0 has the vulnerability CVE-2001-0506 for CodeRed through such simple mapping. Therefore, the system is vulnerable and the decision confidence is increased.

4.2.1.2 Decision Information for Experts

The decisions information of infecting path analysis will be proposed to help experts make correct decisions. Hence, the confirmation effort of experts to decide variant worms could be reduced. Our suggestion is listed as follows.

Suggestion： System Profile matches infecting path of variant worms or not

According to our suggestion, experts will have two decisions.

1) Certain

It is a variant worm and then we will generate knowledge by EMCUD.

2) Uncertain

It is an uncertain case and will be transferred to collaborative analysis center.

In order to clearly show our decision flow, we give an example to illustrate our idea.

Example 4.3：**Uncertain Case in Local VODKA**

The Repertory Grid, AOT and partial embedded rules for Yaha are listed in Example 3.1. Now we have receive inference log as follows：

Table 4.6：**Inference Log from Inference Engine**

Rule #	A_1	A_2	A_3	A_4	A_5	Object	CF
$R_{2,5}$	Email flood	Project.exe	True	{25}	Tcpsvs32.exe	Yaha	0.4
$R_{2,5}$	Email flood	Project.exe	True	{25}	Tcpsvs32.exe	Yaha	0.4
$R_{2,3}$	Email flood	readme.exe	False	{25}	svhoot.exe	Yaha	0.53
$R_{2,5}$	Email flood	Project.exe	True	{25}	Tcpsvs32.exe	Yaha	0.4
$R_{2,5}$	Email flood	Project.exe	True	{25}	Tcpsvs32.exe	Yaha	0.4
$R_{2,3}$	Email flood	readme.exe	false	{25}	svhoot.exe	Yaha	0.53

$R_{2,5}$	Email flood	Project.exe	True	{25}	Tcpsvs32.exe	Yaha	0.4
$R_{2,5}$	Email flood	Project.exe	True	{25}	Tcpsvs32.exe	Yaha	0.4
$R_{2,5}$	Email flood	Project.exe	True	{25}	Tcpsvs32.exe	Yaha	0.4
$R_{2,5}$	Email flood	Project.exe	True	{25}	Tcpsvs32.exe	Yaha	0.4

The large itemset L2=(A_2= Project.exe,A_5=Tcpsvs32.exe) and minimal support 30% is satisfied.

VODKA will ask the following questions

VODKA：Does the attribute-value pair (A_2= Project.exe) AND (A_5=Tcpsvs32.exe) belong to any new variant object?

/*VODKA process in the background*/

Suggestion :

Table 4.7：System Profile of Local Host for Yaha

ID	OS	(Application <-> Vulnerability)
140.113.87.175	WinXp	Outlook<->none

Mapping

Table 4.8：Infecting Path table of Yaha Family

Worm_name	Infecting Path
Yaha	Email；CVE-2001-0154(MIME Header)

The result shows that one of infecting path is not existed. But the normal communication of Email is a path to infect.

VODKA：

Suggestion：System without any vulnerability but Email is the way to infect.

Expert：Uncertain

Experts say uncertain because the context information is not enough and the embedded knowledge with low CF may be caused by improperly operation. The local host may possibly install mass mailer software and the firewall may be temporarily abnormal. Hence, we transfer this uncertain case to the collaborative analysis center.

4.3 Process in Collaborative Analysis Center

Collaborative analysis center will collect uncertain cases from local VODKAs like Example 4.3.

As we know, worm attacking is a network behavior. Therefore, the uncertain case from local VODKA can be clearly explained in collaborative analysis center. If we collect more evidences, we have more confidence to recognize a new variant worm.

4.3.1 The Framework of Collaborative Analysis Center

The framework of collaborative analysis center for computer worms is listed as follows. It consists of three stages, Evidence Collecting Stage, *Worm Analysis Stage* and Knowledge Updating Stage.

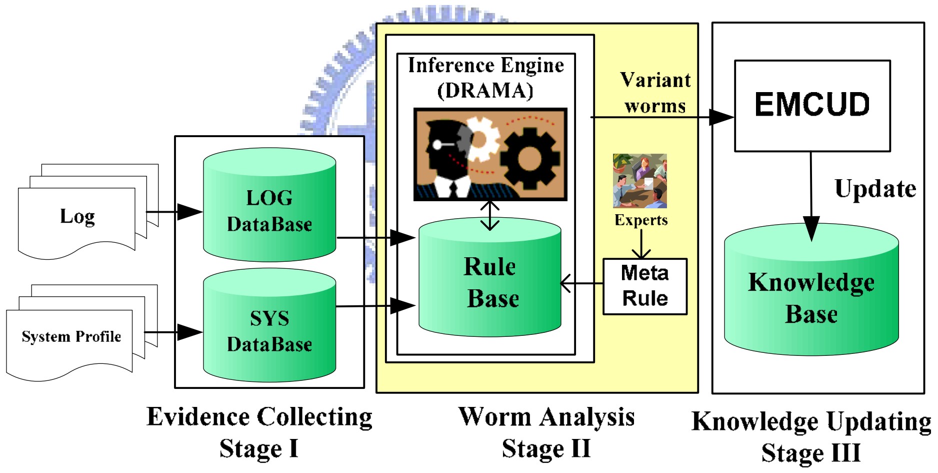

Figure 4.5 : Stage of Global Analysis Center for Computer Worms

Stage I：Evidence Collecting

The Log and System Profile are transferred to our collaborative analysis center directly. The log from each VODKA is reported periodically to collaborative analysis center. However, System Profile would not be changed periodically. We receive the report of System Profile when the configuration of local host is changed. We have the following format of log and system profile.

Table 4.9：Log Profile of Yaha

Time	ID	$R_{i,j}$	A_1	A_2	A_3	A_4	A_5	$CF(R_{i,j})$
T1	140.113.87.175	$R_{2,5}$	Email flood	Project.exe	True	{25}	Tcpsvs32.exe	0.4

Table 4.10：System Profile for Computer Worms

ID	OS	(Application <-> Vulnerability)
140.113.87.175	Windows XP	RPC<->CAN-2003-0352

Figure 4.6 shows the communication between local and collaborative analysis center. And the files are transferred asynchronously. Log is reported periodically and system profile is event-driven.

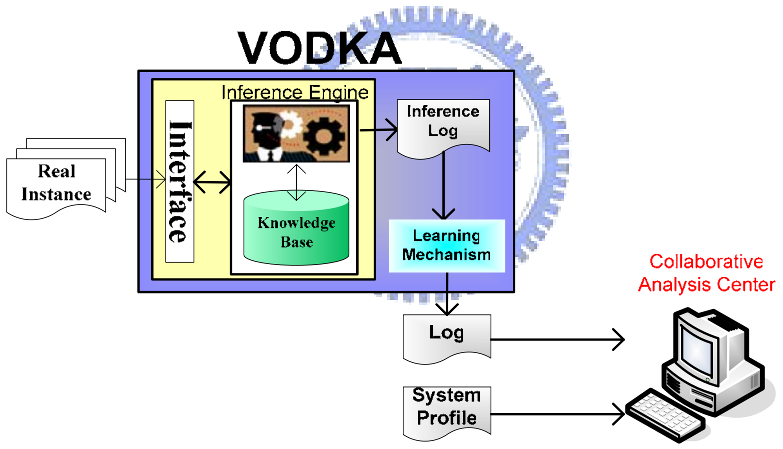

Figure 4.6：Communication between Local and Collaborative Analysis Center

StageII：Worm Analysis Stage

In this Stage we will use expert system to help us systematically analyzing reports. It also can reduce the confirmation effort from experts. We apply extended VODKA to worm domain and we can add more criterions based on the nature of worms. In addition to three basic criterions described in Chapter 3, we know that worms will continually infect other machines, and infecting is the nature for them. So we can add a new criterion for worms about single or multiple reports. If we only receive reports of single host, we do not have enough evidence to verify a variant worm. Of course, domain experts may have their decision ideas about to discover variant worms. At this time they can dynamically add more criterions to fit their requirements.

Now we describe the additional criterion for worm domain.

Additional criterion : Single or multiple reports

As mentioned above, worm attack is network behavior. The reports of variant worms would not be only one. If we only receive reports of single host, our decision confidence is the same as local host. It means the confidence will not be accumulated. Therefore this criterion is to know the reports to accumulate our confidence are from single host or multiple hosts.

Notation :

Object.ID : the IP of this report

Object.HostIP : the IP set of reporting the same object

$\text{Flag}_{\text{object.host}}$: Initialized to 0, set to 1 when receiving different IP

```
MR: IF Object.ID <> Object.HostIP
    Then set $\text{Flag}_{\text{object.host}}=1$
```

According to this criterion, we must modify Criterion3 to correctly discover variant worms. It means if reports from only one host are not considered.

We add **“$\text{Flag}_{\text{object.host}}=1$”** to fit our requirement.

```
MR5: IF $\text{Conf}_{\text{object}}$ > VOthreshold AND $\text{Flag}_{\text{object.host}}=1$
    Then discover variant worms
```

StageIII : Knowledge Updating Stage

When we discover a variant worm, we will acquire the attribute and attribute values and generate relative Repertory Grid and AOT. And feed them to EMCUD to generate defending rules. At the same time we will update our knowledge base.

4.3.2 Run through An Example

In this example, Yaha is an uncertain case described in Example 4.3. Besides Yaha, we also receive reports of Sobig and Gaobot. The example will trace how to analyze with collaborative rules.

Example 4.4 : An Example to Trace Collaborative Analysis Rules

Table 4.11 is the repertory grid of Yaha, Sobig and Gaobot.

Table 4.12 is the AOT of Yaha, Sobig and Gaobot.

Table 4.13 is the partial detecting rules generated by EMCUD.

Table 4.11 : The Repertory Grid of Yaha, Sobig and Gaobot

Object / Attribute	Yaha (O_1)	Sobig(O_2)	Gaobot(O_3)
DoS_Type (A_1)	Email Flood	Email flood	X
Backdoor (A_2)	X	True	X
Email_Attachment(A_3)	Loveletter.doc.pif	Sample.pif	X
Antivirus_Firewall_Abnormal(A_4)	True	X	True
TCP_Port (A_5)	{25}	{25}	{80,135,445}
New_File(A_6)	<****>b.dll	Winmgm32.exe	Bla.exe

Table 4.12 : The AOT of Yaha, Sobig and Gaobot

Object / Attribute	Yaha (O_1)	Sobig(O_2)	Gaobot(O_3)
A_1	D	D	X
A_2	X	3	X
A_3	1	2	X
A_4	3	X	D
A_5	3	3	D
A_6	2	1	1

Table 4.13 : Partial Detection Rules Generated by EMCUD

Rule #	IF Part						Then Part	CF
	A_1	A_2	A_3	A_4	A_5	A_6	Object	
$R_{1,0}$	Email flood	-	Loveletter.doc.pif	True	{25}	<****>b.dll	Yaha	0.8
$R_{1,1}$	Email flood	-	┐Loveletter.doc.pif	True	{25}	<****>b.dll	Yaha	0.7
$R_{1,2}$	Email flood	-	Loveletter.doc.pif	True	{25}	┐<****>b.dll	Yaha	0.6
$R_{1,3}$	Email flood	-	Loveletter.doc.pif	True	{25}	<****>b.dll	Yaha	0.5
$R_{1,4}$	Email flood	-	Loveletter.doc.pif	False	{25}	<****>b.dll	Yaha	0.4
$R_{2,0}$	Email flood	True	Sample.pif	-	{25}	Winmgm32.exe	Sobig	0.8
$R_{2,1}$	Email flood	True	Sample.pif	-	{25}	┐Winmgm32.exe	Sobig	0.7
$R_{2,2}$	Email flood	True	┐Sample.pif	-	{25}	Winmgm32.exe	Sobig	0.6
$R_{2,5}$	Email flood	True	┐Sample.pif	-	{25}	┐Winmgm32.exe	Sobig	0.4
$R_{3,0}$	-	-	-	True	{80,135,445}	Bla.exe	Gaobot	0.7
$R_{3,1}$	-	-	-	True	{80,135,445}	┐Bla.exe	Gaobot	0.6
$R_{3,2}$	-	-	-	True	┐{80,135,445}	Bla.exe	Gaobot	0.5

According to the importance of attributes, we map {D, 5, 4} to major attributes M, {3,2} to secondary attributes S. After mapping, we can get Table 4.14

Table 4.14 : Mapping Table of Yaha, Sobig and Gaobot

A_1	A_2	A_3	A_4	A_5	A_6	object
M	-	-	M	M	S	Yaha
M	M	S	-	M	-	Sobig
-	-	-	M	M	-	Gaobot

Table 4.15 : System Profile of Three Hosts

ID	OS	Application<->Vulnerability
140.113.87.175	Windows XP	RPC<->CAN-2003-0352
140.113.87.174	Windows 2000	none
140.113.167.101	WindowsXP	RPC<->CAN-2003-0352

Constructing collaborative rules from all criterions

In this example, we will use formula to calculate $Conf_{object}$ of each rule and set ($Conf_{high}$=0.2), ($Conf_{low}$=0.1), ($Conf_{SystemProfile}$=0.3), ($flag_{object}$=0) and ($Conf_{object}$= 0).

Collaborative analysis rules for Yaha

```
IF (DoS=Email Flood) AND (Antivirus Firewall abnormal=True) AND
    (TCP port={25}) AND (New file=svchook.dll)
    Then  ($Conf_{Yaha}$= $Conf_{Yaha}$ + 0.2) AND ($flag_{Yaha}$=1)
IF ($flag_{Yaha}$=0) AND (DoS=Email Flood) AND
    (Antivirus Firewall abnormal=True) AND (TCP port={25})
    Then  ($Conf_{Yaha}$= $Conf_{Yaha}$ + 0.1)
IF ($flag_{Yaha}$=1) Then  ($flag_{Yaha}$=0)
```

Collaborative analysis rules for Sobig

```
IF (DoS=Email Flood) AND (Backdoor=True) AND (TCP port={25}) AND
    (Attachment=Sample.gif)
    Then  ($Conf_{Sobig}$= $Conf_{Sobig}$ +0.2) AND ($flag_{Sobig}$=1)
IF ($flag_{Sobig}$=0) AND (DoS=Email Flood) AND (Backdoor=True) AND
    (TCP port={25})
    Then ($Conf_{Sobig}$= $Conf_{Sobig}$ +0.1)
IF ($flag_{Sobig}$=1) Then  ($flag_{Sobig}$=0)
```

Collaborative analysis rules for Gaobot

```
IF (Antivirus Firewall abnormal=True) AND (TCP port={80,135,445})
    Then $Conf_{Gaobot}= Conf_{Gaobot} +0.1$
IF ID.SystemProfile = { CAN-2003-0352 ; CAN-2003-0003}
    Then $Conf_{Gaobot}= Conf_{Gaobot} + 0.3$
```

Collaborative rules for discovering variant worms

Initialize $flag_{object.host}$=0

```
IF ($Conf_{Yaha}$>=0.8) AND ($flag_{Yaha.host}$=1)Then (VariantWorm=Yaha)
IF ($Conf_{Sobig}$>=0.8) AND ($flag_{Sobig.host}$=1)Then (VariantWorm=Sobig)
IF ($Conf_{Gaobot}$>=0.8) AND ($flag_{Gaobot.host}$=1)Then (VariantWorm=Gaobot)
```

The flag, $flag_{object.host}$ means the confidence is accumulated from single host(=0) or multiple host(=1).

Collaborative rules for timeout period

```
IF (Timeperiod=3T)
    Then ($Conf_{Yaha}$= 0.5 * $Conf_{Yaha}$) AND ($Conf_{Yaha}$= 0.5 * $Conf_{Yaha}$) AND
            ($Conf_{Yaha}$= 0.5 * $Conf_{Yaha}$)
```

After generating collaborative analysis rules of Yaha, Sobig and Gaobot, the inference engine is driven with reports from local hosts.

Data collecting

The reports are collected from local VODKAs (Take three hosts as examples).

We select useful attributes from log database to trace our example. And they are Time, ID and six attributes.

Event	Time	ID	A1	A2	A3	A4	A5	A6
E1	T1	140.113.87.175	Email flood	-	Ravs.scr	T	{25}	WinServices.exe
E2	T1	140.113.87.174	Email flood	-	Patch.exe	T	{25}	-
E3	T1	140.113.167.101	Email flood	T	Password.gif	-	{25}	mscch32.exe
E4	T2	140.113.87.175	-	-	-	T	{80;135;445}	msgconf.exe
E5	T2	140.113.87.174	Email flood	T	Peace.scr	-	{25}	WinServices.exe
E6	T2	140.113.167.101	-	-	-	T	{80;135;445}	MSRUN.exe

Now, we trace each report to see the change of confidence.

E1：matching first collaborative rules of Yaha

$Conf_{Yaha}=0+0.2=0.2$

E2：matching second collaborative rules of Yaha

$Conf_{Yaha}=0.2+0.1=0.3$

E3：matching first collaborative rules of Sobig

$Conf_{sobig}=0+0.2=0.2$

E4：matching collaborative rules of Gaobot

$Conf_{Gaobot}=0+0.2=0.2$

This case will consider system profile of local host

We can discover E4 has vulnerability.

We increase $Conf_{Gaobot}=0.2+0.3=0.5$

E5：matching first collaborative rules of Yaha

$Conf_{Yaha}=0.3+0.2=0.5$

E6：matching collaborative rules of Gaobot

$Conf_{Gaobot}=0.5+0.2=0.7$

$Conf_{Gaobot}=0.7+0.3=1.0 > 0.8$ => Alert Gaobot Variant

Knowledge updating phase

When discovering a new variant, we generate the new acquisition table of Gaobot.B.

Table 4.16 : The Acquisition Table of Gaobot

Object / Attribute	Gaobot.B
Antivirus_Firewall_Abnormal(A_4)	True
TCP_Port (A_5)	{80,135,445}
New_File(A_6)	{msgconf.exe; MSRUN.exe}

Hence, an original rule and and embedded rule are listed as follows.

"**IF** (Antivirus Firewall abnormal=True) **AND** (TCP port={80,135,445}) **AND** (New file={msgconf.exe; MSRUN.exe}) **THEN** Gaobot.B CF=0.7

"**IF** (Antivirus Firewall abnormal=True) **AND** (TCP port={80,135,445}) **AND** ┐(New file={msgconf.exe; MSRUN.exe}) **THEN** Gaobot.B CF=0.4

4.4 Evaluation

4.4.1 Experiment Environment

In the experiment, we deployed Extended VODKA to eight WISE sensors located in two subnets and a collaborative analysis center with expert system helps us analyzing local uncertain cases systematically. The experiment environment is shown as Figure 4.7.

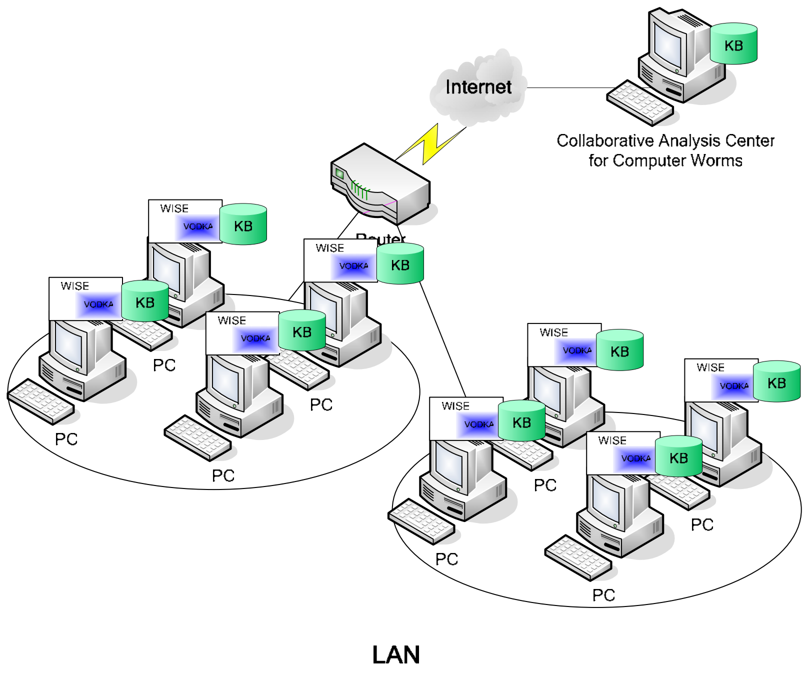

Figure 4.7：Experiment Environment

Before the expert system is activated, we must set parameters listed as follows:

Conf_{high} : 0.2

Conf_{low} : 0.1

$\mathrm{Conf}_{SystemProfile}$: 0.15

TH_{period} : 30 minutes of half-life period

$\mathrm{VO}_{threshold}$: 0.8 of alert threshold

After that, in order to verify that our collaborative defending system can discover variant worms effectively and precisely from VODKAs, the three kinds of datasets are used including **Baseline**, **Normal with some legitimate software** and **Attack**.

A. Baseline

The dataset is gained from the eight sensors when the system is initially installed without any heavy load of software.

B. Normal with some legitimate software

The dataset is gained after some software like P2P, FTP, email software, etc. are installed. They might generate doubtful data temporarily like worms.

C. Attack

The dataset is gained after a new variant worm is inserting and attacking our machines.

4.4.2 Results

The result presents the variation of confidence as time goes on. This can observe a variant worm appear or not when the confidence exceeds a predefined threshold 0.8.

A. Baseline

From Figure 4.8, we can observe the variation of confidence is quite slight.

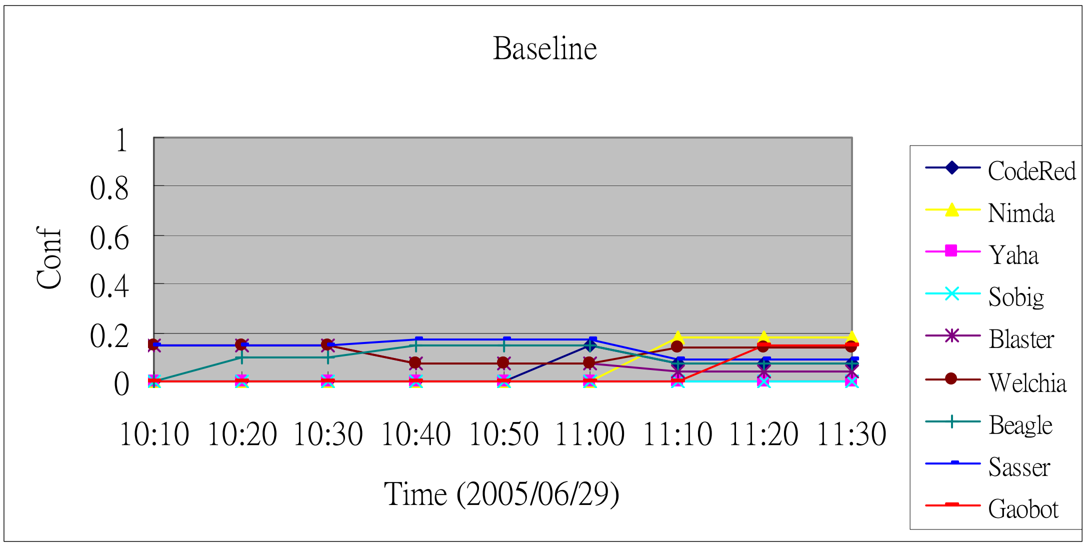

Figure 4.8：The Variation of Confidence of Dataset Baseline

B. Normal with some legitimate software

In the normal state, the dataset is generated by some legitimate software. And then we might observe the variation of confidence of nine worms. During the period of 90 minutes, no alert is triggered by our system.

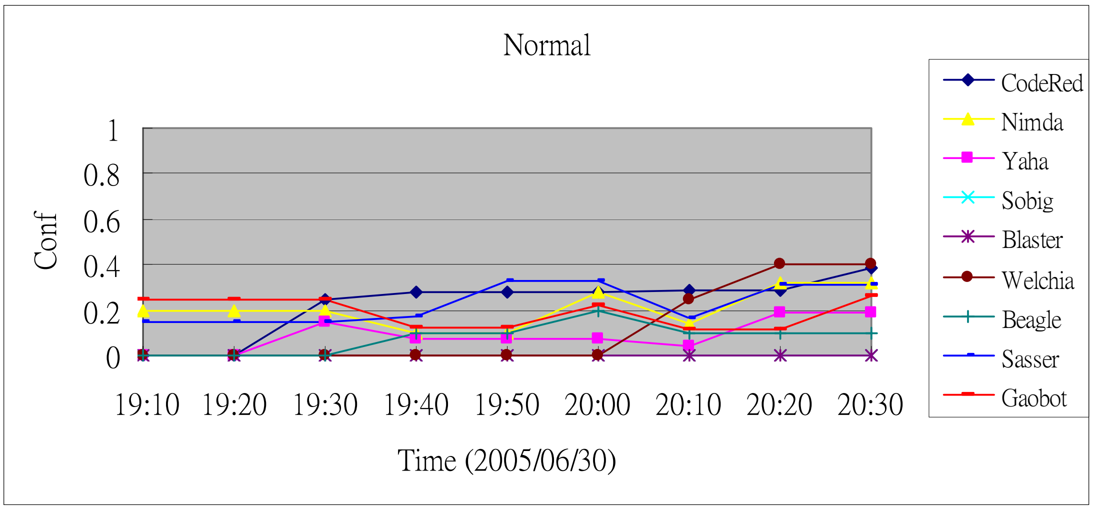

Figure 4.9：The Variation of Confidence of Dataset Normal

C. Attack

When a new variant worm which is not existed in our knowledge base at 13:20 was released, we found the confidence value of Gaobot increasing rapidly. Besides one sensor is infected by Gaobot, the worm will also infect other system with the same vulnerability. This will cause the confidence value to increase rapidly. At 13:50, a variant worm of Gaobot is discovered by our system and then we obtain frequent itemset to ask experts to confirm such itemset. Finally, EMCUD is used to generate the defending rules for the variant worm of Gaobot.

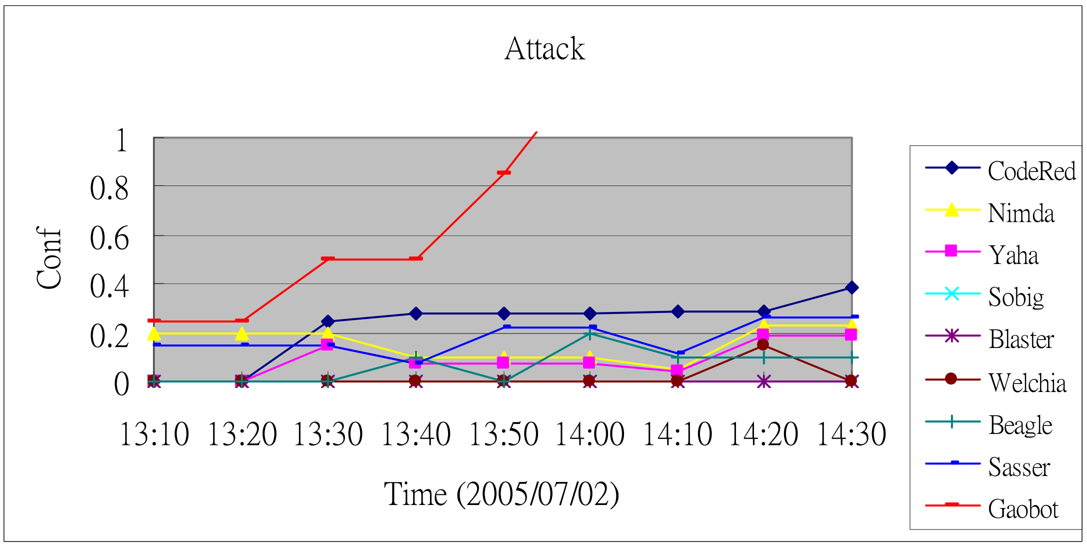

Figure 4.10：The Variation of Confidence of Dataset Attack

4.4.3 Discussion

The correctness of Collaborative Defending System

In order to test the robustness of our criterions, three kinds of datasets are used to test the reaction ability of collaborative defending system.

In Figure 4.8, the variation of confidence is less accumulated by the dataset of baseline. The result is clear and system work well. However, it is not enough to test the correctness of our criterions. Hence, the dataset of Normal with some normal software, generated by installing P2P, FTP, E-mail software, etc., can observe the false alarm rate of our criterions. Although the dataset might suddenly generate doubtful attack data temporarily, the result of Figure 4.9 shows that the system is reacted accurately. However, this does not prove our system can discover variant worm properly. Hence, we insert a variant worm, Gaobot to test the correctness and reaction time. In Figure 4.10, the result shows our system can detect the new variant worm, Gaobot immediately.

The influence of parameters

Theoretically speaking, the value of parameters will affect the alert time and the discovery frequency. Hence, the confidence of each rule must be well defined. For example, if they are higher, large number of alerts may cause experts much effort. If they are lower, variant worms would be missed or generate alerts slowly. In the experiment, the value of parameters in the experiment environment would be defined higher than in the real world for increasing the reaction time of the fewer sensors. In order to reduce the false alarm of higher value, we reduce the half-life time to thirty minutes. Therefore, we can avoid the confidence value increasing unlimitedly and reduce some noise.

Chapter 5. Conclusion and Future Work

In the thesis, we propose a methodology to extend VODKA to help domain experts discover variant objects easily. First, to reduce the conformation effort of experts, we extend VODKA to consult more context information, such as environment factors, for helping experts make the correct decision. Although more information is provided, experts may still feel uncertain to confirm the occurrence of variant object in local environment. And then a collaborative analysis framework is proposed to handle uncertain cases for further enhancement. The collaborative analysis framework is a systematical analysis mechanism based upon expert system to discover variant objects according to the meta knowledge including environment factors and domain specific heuristic criteria.

Furthermore, the defending system for computer worms is implemented to evaluate our proposed extended VODKA. According to the variation of confidence, we can easily solve the local uncertain cases using the collaborative analysis rules.

In our knowledge base, we can only learn parts of worms in real world. However, with the rapid growth of variant worms, we need to enrich our knowledge base to discover more variant worms. More defending knowledge can help to reduce the threat posed by worms.

References

[1] P. Crowther and J. Hartnett,"Using repertory grids for knowledge acquisition for spatial expert system" Proceeding on In Intelligent Information System, November 1996.

[2] A. Dixit and E. Pindyck, "Investment under uncertainty." Princeton NJ. Princeton University Press, 1994.

[3] Dan Ellis, McLean, "Worm Anatomy and Model." Proceedings of the ACM workshop on Rapid Malcode, 2003

[4] Chien Eric. "Code Red Worm." Symantec Security Response, July 2003, http://securityresponse.symantec.com/avcenter/venc/data/codered.worm.html

[5] Chien Eric. "W32.Nimda.A@mm." Symantec Security Response, July 2003, http://securityresponse.symantec.com/avcenter/venc/data/w32.nimda.a@mm.html

[6] G. J. Hwang and S.S. Tseng, "On building a medical diagnostic system of acute exanthema." Journal of Chinese Institute of Engineers, Vol. 14, No. 2, 185-195,1991

[7] 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.

[8] D.M. Kienzle, M. C. Elder, "Recently worms: a survey and trends." WORM'03, October 2003

[9] Douglas Knowles. "W32.Gaobot.Worm." Symmantec Security Response, October 2002, http://securityresponse.symantec.com/avcenter/venc/data/w32. hllw.gaobot.html

[10] Douglas Knowles. "W32.Sobig.Worm." Symantec Security Response, January 2003, http://securityresponse.symantec.com/avcenter/venc/data/w32.sobig.a@mm.html

[11] Douglas Knowles. "W32.Yaha.Worm." Symmantec Security Response, January 2004, http://securityresponse.symantec.com/avcenter/venc/data/w32.yaha@mm.html

[12] F. Lau, S.H. Rubin, M.H. Smith, L. Traikoyic, "Distributed Denial of Service Attacks", 2000 IEEE International Conference on Volume 3, 8-11 Oct. 2000 Page(s):2275 - 2280 vol.3

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

[14] S.S. Tseng, S.C. Lin and L.H. Liu, “VODKA: Variant Objects Discovering Knowledge Acquisition”, submitted to IJHCS Oct. 2004.

[15] Nicholas Weaver, Vern Paxson, Stuart Staniford, Robert Cunningham , “A Taxonomy of Computer Worms”, Proceedings of the 2003 ACM workshop on Rapid Malcode, October 2003