An approach to fast malware classification with machine learning technique

Pham Van Hung

Faculty of Environment and Information Studies

Keio University

5322 Endo Fujisawa Kanagawa 252-0882 JAPAN

Submitted in partial fulfillment of the requirements

for the degree of Bachelor

Advisors:

Professor Hideyuki Tokuda

Professor Jun Murai

Associate Professor Hiroyuki Kusumoto

Professor Osamu Nakamura

Associate Professor Kazunori Takashio

Assistant Professor Rodney D. Van Meter III

Associate Professor Keisuke Uehara

Associate Professor Jin Mitsugi

Lecturer Jin Nakazawa

Professor Keiji Takeda

Abstract of Bachelor Thesis

An approach to fast malware classification based on malware's meta-data using machine learning technique

With the rapid increase of malware, it is important for malware analysis to classify unknown malware files into malware families. From that, the behavior and characteristics of malware will be characterized accurately. In this paper, an approach was introduced for perform fast malware classification based on meta-data of malware's file. A machine learning technique, called decision tree algorithm, is used to classify malware rapidly and correctly. Experimental results of the malware samples show that our system successfully determined some semantic malware similarities, especially showed their inner similarities in behavior and static malware characteristic.

Keywords:
Malware, static analysis, decision tree.

Pham Van Hung

Faculty of Environment and Information Studies

Keio University

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Back ground

Malware is general word for all type of malicious software. Malware includes Virus, Trojan house, Back door, Worm, and other malicious software which are characterized by malicious code. Because of the widespread use of the Internet, computer user face many dangerous propagation of malware. The modern malware purpose is usually intended to gain illegal profit. For example, million computers infected million total number of American user computers for e-payment system losing $24.3 million[3]. In addition, According to 2010 Annual Security Report, on May 2010, tens of millions computer world wide was infected by email worm such as I LOVE YOU, LOVE LETTER, LOVEBUG[2]. As a result, preventing, detecting, and removing malware is very important for security networking.

1.2 Malware analysis problem

Instead of effective preventing, detecting, removing, and analyzing malware is analysed to find common areas that all viruses in a family share uniquely, and can thus create a set of signature in order to detect malwares. Commonly, when new malware was detected, dynamic malware analysis and static malware analysis have used to analyze the malware. Dynamic malware analysis technique that execute malware in the Virtual Machine and use ProcMon, RegShot, and other tools to identify the general behavioral analysis techniques such as network traffic analysis; file system; and other Window feature: service, process, the registry. Unfortunately, these dynamic techniques are susceptible to a variety of anti-monitoring defenses, as well as *time bombs* or *logic bombs* and can be slow and tedious to identify and disable code analysis techniques to unpack the

code for examination [36]. Furthermore, it takes large amount of time to prepare environment to analyze malware such as virtual machine environment but some malware can not be executed in virtual machine environment. With the static malware analysis technique, researcher perform reverse engineering using IDA Pro and Ollydbg tool to analyze malware by seeing the structure of malware, in order to discover its purpose and functionality but it takes a lot of time to see the malware structure. Malware analysis is necessary to understand the behavior of malware. Therefore, malware signature is created to detect malware effectively. However, With a vast amount of sample increased day by day, anti-virus industry and virus researchers is harder to analyze malware without information of new malware. With the aim of reducing time of malware analysis, it is necessary to have an automatically malware classification system.

1.3 Approach

Two following issues are focused on:

- Automatically perform fast malware classification based on malware file's meta-data using a machine learning technique, called decision tree algorithm to classify unknown malwares or subspecies rapidly and correctly.
- Help researcher to understand which family malware belongs to and detect some semantic information about malware so they can make.

For that reason, in this paper, we propose an approach to perform fast malware classification based on malware's meta-data using machine learning technique, known as decision tree.

1.4 Thesis outline

The rest of this thesis is structured as follows:

- Chapter 2 describes malware meta-data, and the method in static classification of malware. In sight is provided the purpose of method given in this research.
- Chapter 3 presents the other malware static classification approach.
- Chapter 4 approach, and the design and operation of malware classification system is described.

- Chapter 5 presents environment and implementation of static malware system.
- Chapter 6 describes the system evaluation method and results.
- Summary, the conclusion and future work is presented in chapter 7

Chapter 2

Back ground

This chapter presents the growth of malware. Afterwards, malware analysis technique is introduced. Afterwards, malware categories and the way to use Virus total service detect malware name that defined by anti-virus vendor is introduced. Finally, the problems of using malware name to detect semantic meaning of malware.

2.1 Growth of malware attack

Malware can self-replicate recursively. For example, Mota is a kind of worm that spread itself by sending spam email to address that harvested local machine. In addition, Downadup is a malware that receive and execute file through a peer-to-peer systems by communication between computers. Malware infects system and spread to the other systems by communication tools. Recently, malware infects system and send the copy of itself to the others systems by Internet. Since the rise of widespread broadband Internet access, the number of malware samples has rapidly increased. According to the report issued by Kaspersky's research team, 205 million pieces of malware were detected and neutralized [27]. In addition, in 2010 the number of malware samples increased by 20 millions [7].Figure 2.1 show the increasing of malware from 2003 to 2009 by Kaspersky Lab. As a result of the rapid growth malware, malware is still a huge problem in security internet and network connectivity.

At this time, malware is easy created by Malware Creation Tool, as know as a program that used by attacker to generate malware[4]. In addition, unlike early attack tools that implement one type of attack, such tools now can be changed quickly by upgrading or replacing their code. This causes rapidly evolving attacks and, at the extreme, results in polymorphic tools that self-evolve, changing with each active instance of the attack. Therefore, a large amount of malware has challenged

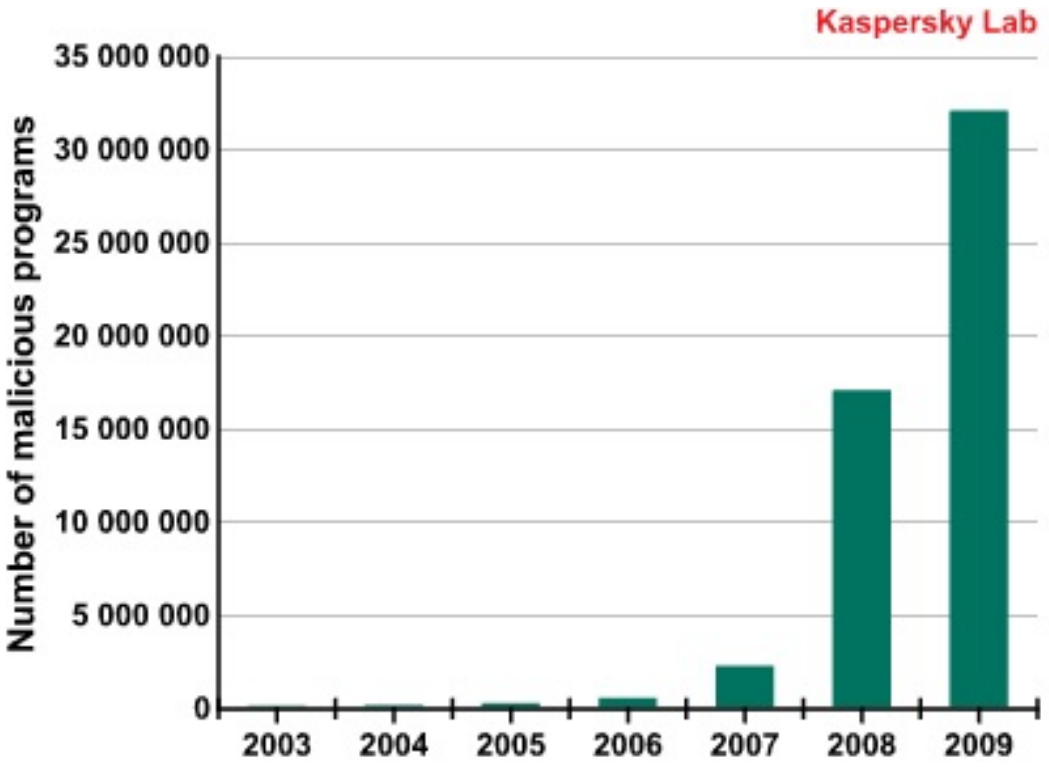

Figure 2.1: Number of malicious program

[7]

anti-virus vendor and researcher in effective analysis.

2.2 Malware avoidance technique

At the present time, malware is implemented with avoidance technique in order to invalidate static signature based method by anti-virus software and make analysis process more complicated. Avoidance technique can change malware signature and syntax without the changing the behaviors of malware. The avoidance technique consists of obfuscating code and packing technique. There fore, a avoidance technique make malware analysis more complicated.

Obfuscating code change the form of instance malware to other form in order to invalidate signature based detection technique. Obfuscating code consists of polymorphism and metamorphism [15]. Polymorphic technique change the representation of malware. Virus, worm, and other self-propagating software if often used polymorphic technique such as encryption, data appending, and data pre-pending. Metamorphic malware automatically recode it self when it distributed or executed[15]. Simple metamorphic techniques include: adding varying lengths of NOP instructions,

adding useless instruction, and loop the code segment. Advantage metamorphism techniques include: function reordering, program flow modification, static structure modification.

Packing malware can compress the Win32 portable execution file by several tool such as UPX. According to a study carried out by Panda Research, 78% of new malware used some kind of file packing to evade detection. PE-packer is designed to reduce the size of malware through compression. The size of packed malware is small but it is bigger when it run in system[17]. When uncompressed in memory, packed malware is normally executed. UPX and some PE-packer compress malware binary files and make analysis malware more harder.
For the reason that modern malware is implemented with avoidance technique, detection method by the use of static signatures is criticized for being ineffective.

2.3 Malware analysis technique

This section describes two malware analysis technique includes: dynamic and static analysis. The detail of two techniques is described as follow:

2.3.1 Dynamic malware analysis

Dynamic malware analysis technique is technique to find out the purpose of malware by run it and make sure what happen in system. In general, malware is executed in the Virtual Machine. After that, malware researcher use SysAnalyzer, Process Explorer, ProcMon, RegShot, and other tools to identify the general behavioral analysis techniques. For example, SysAnalyser is useful analysis tool to monitors many aspects of system and process states such as running process, open ports, loaded drivers, injected libraries, key registry changes, APIs called by a target process, file Modifications, http, IRC, and DNS traffic. Figure 2.2 show the example of using SysAnalyser tool to analyse malware. The advantage of Dynamic malware is easy to detect malware behavior. Therefore, researcher can remove malware effectively. However, these dynamic techniques are susceptible to a variety of anti-monitoring defenses, as well as *time bombs* or *logic bombs* and can be slow and tedious to identify and disable code analysis techniques to unpack the code for examination [36]. Furthermore, it takes large amount of time to prepare environment to analyze malware such as virtual machine environment but some malware cannot be executed in virtual machine environment.

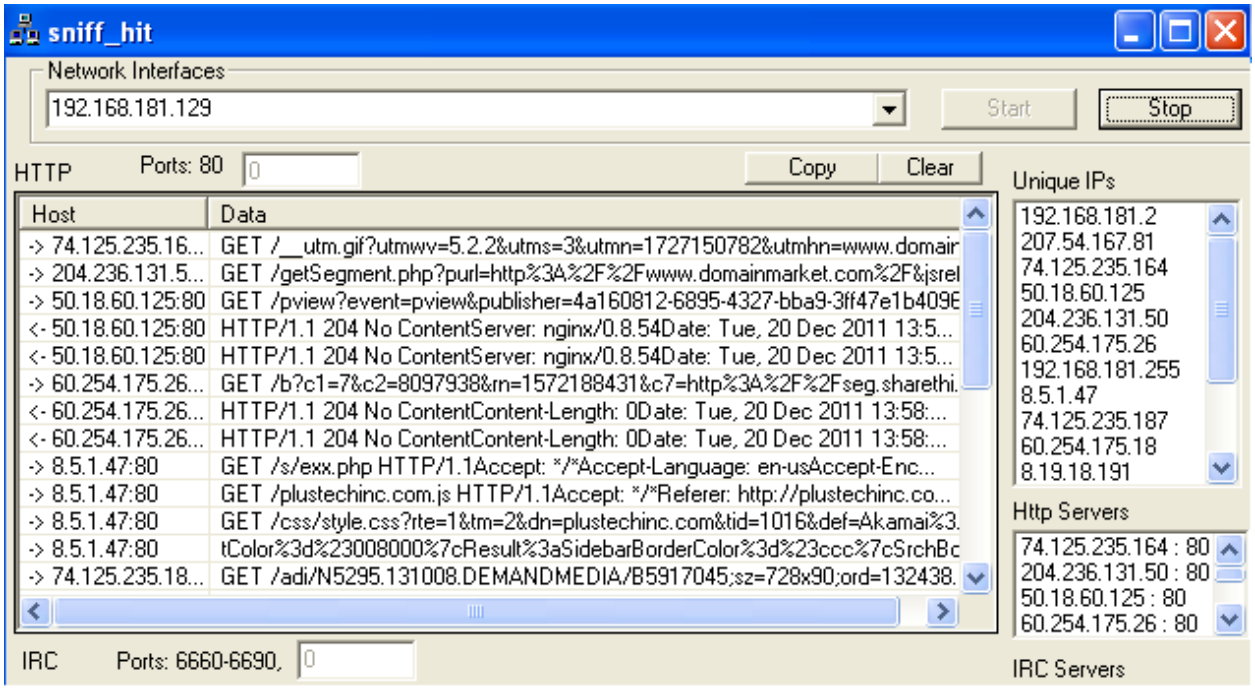

Figure 2.2: SysAnalyser tool for dynamic analysis.

2.3.2 Static malware analysis

Static malware analysis is technique that identify malware program without executing it. With the static malware analysis technique, researcher performs reverse engineering using disassemble tools, decompile tools, source code analyzer tools such as IDA pro and Ollydbg in order to understand malware by seeing the structure of malware. Static malware analysis have an advantage that it can completely discover malware purpose and functionality. However, research takes many time to understand the malware functionality by analysing malware structure.
Figure 2.3 show the example of using OllyDbg tool in order to analyse malware.

2.4 Malware categories

In general, malware is classified into a few categories or types based on behaviors, method of infection, and the resulting propagation of malware. For example, this is some malware categories: virus, trojan, worm, spyware, and rookit. The categories detail specific types of malware threats:

- Virus :"Software which infects other application and use them as a spreading medium"[25].
- Trojan :"A malicious application which present itself as something else"[25].
- Worm :"Code with ability to spread from computer to computer by means of different network

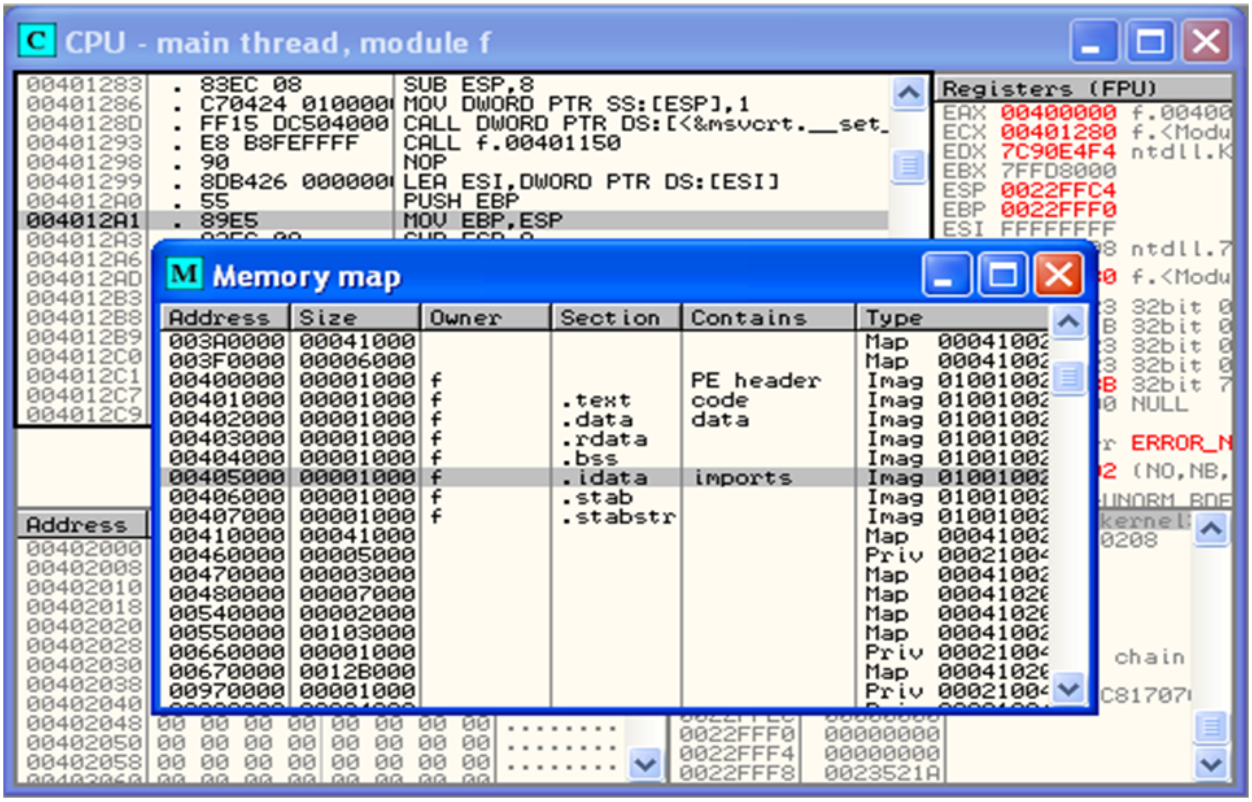

Figure 2.3: Ollydbg tool for static analysis.

protocols"[25].

- Spyware :"Application aiming to havest personal information"[25].
- Rookit :"Hidden tools providing stealth services to its writer"[25].

However, As the different between the categories are a bit fuzzy, and the classes are obviously not exclusive. In addition, depend on purpose of virus industry a unique malware can belong to rookit, virus, or spyware. The detail of malware categories is presented on the next section.

2.4.1 Use virus total to detect the name of categories.

In this thesis, MD5 hash is used to detect malware name which provided by many anti-virus vendor. An MD5 hash is generated by MD5 Message-Digest Algorithm, is a widely used cryptographic hash function that produces a 128-bit (16-byte) hash value. An MD5 hash is typically expressed as a 32-digit hexadecimal number, and MD5 is not collision resistant[24].

2.4.2 Using virus total to getting vendor name

In this paper, MD5 hash of malware is used to search the name of malware by virus total. Virustotal is a web service that analyzes malware files and facilitates quick detection of viruses, worms, trojans, and all kinds of malware detected by antivirus engines. Malware's name is provided by various anti-virus vendor, but there are many name for unique malware. The Figure 2.4 shows malware name detected by antivirus engines.

Antivirus	Version	Last Update	Result
AhnLab-V3	2011.08.13.00	2011.08.13	Win-Trojan/Downloader.6144.QR
AntiVir	7.11.13.37	2011.08.12	TR/Crypt.XPACK.Gen
Antiy-AVL	2.0.3.7	2011.08.13	Trojan/Win32.Agent.gen
Avast	4.8.1351.0	2011.08.14	Win32:Trojan-gen
Avast5	5.0.677.0	2011.08.14	Win32:Trojan-gen
AVG	10.0.0.1190	2011.08.14	Win32/Heur
BitDefender	7.2	2011.08.14	Trojan.Generic.304464
CAT-QuickHeal	11.00	2011.08.13	TrojanDownloader.Agent.pkw
ClamAV	0.97.0.0	2011.08.13	Trojan.Downloader-38041
Commtouch	5.3.2.6	2011.08.13	W32/Downldr2.BXKD
Comodo	9739	2011.08.14	TrojWare.Win32.TrojanDownloader.Small.OBW
DrWeb	5.0.2.03300	2011.08.14	Trojan.DownLoader.61643
Emsisoft	5.1.0.8	2011.08.14	Trojan-Dropper.Agent!IK
eSafe	7.0.17.0	2011.08.10	Win32.Agent.pkw
eTrust-Vet	36.1.8499	2011.08.12	-
F-Prot	4.6.2.117	2011.08.13	W32/Downldr2.BXKD
F-Secure	9.0.16440.0	2011.08.14	Trojan.Generic.304464
Fortinet	4.2.257.0	2011.08.14	W32/Agent.PKW!tr.dldr
GData	22	2011.08.14	Trojan.Generic.304464
Ikarus	T3.1.1.107.0	2011.08.14	Trojan-Dropper.Agent
Jiangmin	13.0.900	2011.08.13	TrojanDownloader.Agent.aavi
K7AntiVirus	9.109.5010	2011.08.12	Trojan-Downloader
Kaspersky	9.0.0.837	2011.08.14	Trojan-Downloader.Win32.Agent.pkw

Figure 2.4: Malware name is detected by antivirus engines.

2.5 Problems of malware name

As the malware name detected by anti-virus engines show in Figure 2.4, unique malware is not classified into unique name. For each anti-virus vendor, the name of malware detected is different to

the other anti-virus vendor. The classification of each anti-virus engine is unlike others. Therefore, malware detected by anti-virus engines cannot provide meaningful characterization of malware for virus researcher. In order to overcome this problem, this paper proposed to classify the malware into families based on malware specific target and its operation behavior. As a result of malware name detected by anti-virus vendor was presented above, the new malware families is required for detect meaningful characterization of malware.

2.6 Malware families is used in this paper

This paper use malware families which is reported by Information-technology Promotion Agency [8]. The table show malware families used in this paper such as Win32/Virut, Win32/Autorun, Win32/IRCbot, Win32/Gaobot, Win32/Waledac, Win32/Downadup, Win32/Sality, and W32.Mota.

Malware familes	Summary
Win32/Virut	"Win32/Virut is a family of file infecting viruses that target and infect .EXE and .SCR files accessed on infected systems. Win32/Virut also opens a backdoor by connecting to an IRC server, allowing a remote attacker to download and run files on the infected computer." [12]
Win32/Autorun	"Win32/Autorun is a family of worms that spreads by copying itself to the mapped drives of an infected computer. The mapped drives may include network or removable drives." [11]
Win32/IRCbot	"Win32/IRCbot is a large family of backdoor Trojans that targets computers running Microsoft Windows. The Trojan drops other malicious software and opens a backdoor on the infected computer to connect to IRC servers. The Trojan can maintain multiple IRC server connections simultaneously to receive commands from attackers." [10]
Win32/Gaobot	"The Win32/Gaobot worm family spreads using different methods, depending on the variant. Some variants spread to machines with weak passwords. Others exploit vulnerabilities to infect machines. Once a machine is infected, the worm connects to an IRC server to receive commands." [9]
Win32/Waledac	"Win32/Waledac is a trojan that is used to send spam. It also has the ability to download and execute arbitrary files, harvest email addresses from the local machine, perform denial of service attacks, proxy network traffic and sniff passwords." [**?**]
Win32/Downadup	"Win32/Downadup attempts to spread to network shares by brute-forcing commonly used network passwords and by copying itself to removable drives." [**?**]
Win32/Sality	"Virus:Win32/Sality is a family of polymorphic file infectors that target Windows executable files with the extensions .SCR or .EXE. They may execute a damaging payload that deletes files with certain extensions and terminates security-related processes and services.
W32.Mota	W32.Mota is a worm that propagates by sending itself to email addresses gathered from the computer." [21]

Table 2.1: Malware

Chapter 3

Related research

This chapter presents the recent approach for automatically classify malware into malware families. For the reason that the detection based static signature is no longer effective in chapter 2, at this time new malware classification method is required to detect mawlare writers used avoidance technique.

3.1 Flow graph

An another approach is using emulator to automatically unpack the packing malware, then from reverse code produce flowgraph, then flowgraph matching to perform classification[28]. The system created by the approach follows:

- Fist, System automatically unpack malware based on application level emulation, and then use entropy analysis to detect.

- Second, Produce control followgraph by using graph invaiant based signature in order to measure similarity between malware.

- Next, Generic string based control flow signature, in order to using string edit distancce. Automatically unpack malware based on application level emulation.

- Finally, Malware classification use a set similarity function and a set similarity search algorithm to represent benign and malicious classes.

The disadvantage in flowgraph approach is high cost of runtime complexity. Firstly, runtime complexity of of malware classification is $O(N \log M)$ where M is the number of control follow graphs in database, and N is the number of control follow graph input binary [28]. In addition, To identify two flowgraph is N^3, and N is the number of nodes in each graph. Further more, needs to unpack the sample if it was packed with an executable packer. There fore, this approach is ineffective in malware classification system with a large number of instances.
In addition, malware classification based flowgraph approach cannot accurately detect metamorphic malware. As presented at chapter 2, metamorphic malware can recode it self with program follow modification, function reordering. There fore, it is hard to classify malware based a set similarity between follow graph.

3.2 Optimizing decision tree in malware classification system by using Generic Algorithm

In this time, malware classification system can use machine learning classifier. The main idea of machine learning technique classifier is search algorithm that learn from externally supplied instances to produce a concise model of the distribution, which then make prediction about new instances. Current machine learning classifier in malware classification include: Naive Bayse, Suport Vector machine, Decision tree, K-nearest Neighbor [14]. This approach use combining Generic algorithms with Decision tree. The data set is separated in training data set and testing data set. The data set is include:
// Malware target, specific target for malware attack
vector¡list¡MalwareClass¿¿ _slots;

// Malware classes for chromosome
hash_map¡MalwareClass,int¿ _Dataclasses;
hash_map¡MalwareClass,int¿ _Appsclasses;
hash_map¡MalwareClass,int¿ _Sysclasses;
hash_map¡MalwareClass,int¿ _Dosclasses;

Generic algorithm is methods that analogous to the process of natural evolution. Generic algo-

rithm is used to optimize decision tree in order to accurately classify malware. Malware classification system, which implemented by the combining generic algorithm with decision tree algorithm approach, is for classify malware into two classes: benign program and malicious program, not for detect semantic characterization of malware by classifying malware into families. There fore, the combining generic algorithm with decision tree algorithm approach cannot use for detecting semantic characterization of malware.

3.3 Conclustion

To achieve fast malware classification, two approach is described before can not be used in this system. New approach that is for fast malware classification system will introduced in the next chapter.

Chapter 4

Classification based on malware's meta-data using decision tree approach

This chapter describes Win32 executive files structure, known as PE files structure, that used in classification system as malware meta-data. Afterwards, the machine learning technique, known as decision tree method, that used in classification malware system is described. Finally, this chapter presents classification based on malware's meta-data using decision tree approach.

4.1 PE file format

4.1.1 PE file overview

The real content of the PE file is divided into blocks called sections. A section is nothing more than a block of data with common attributes such as code/data, read/write etc. You can think of a PE file as a logical disk. The PE header is the boot sector and the sections are files in the dvisk. The files can have different attributes such as read-only, system, hidden, archive and so on. When the PE loader maps the sections into memory, it examines the attributes of the sections and gives the memory block occupied by the sections the indicated attributes.
If the PE file format is viewed as a logical disk, the PE header as the boot sector and the sections as files, we still don't have enough information to find out where the files reside on the disk, ie. Each structure contains the information about each section in the PE file such as its attribute, the

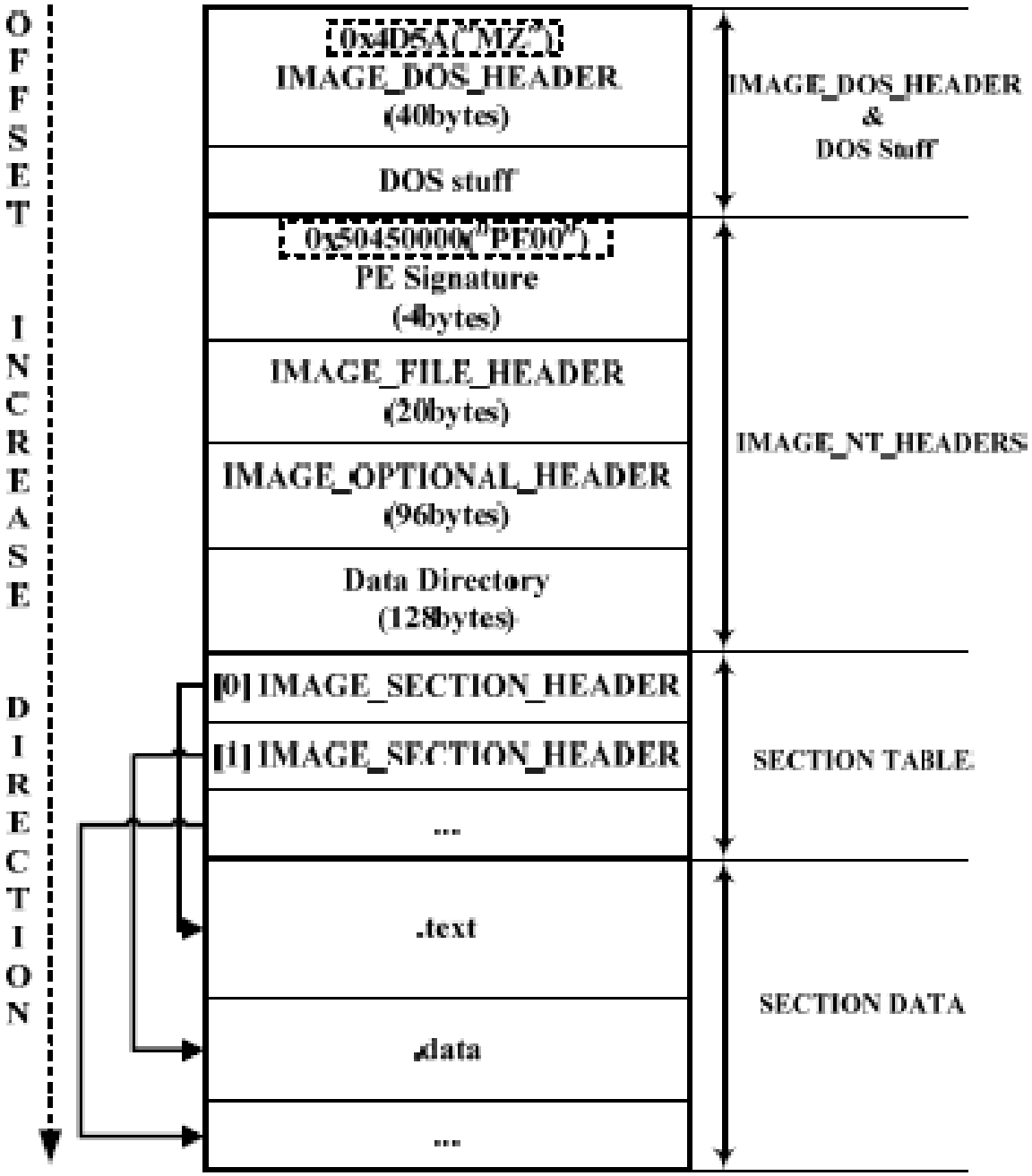

Figure 4.1: PE file format.

file offset, virtual offset. If there are 5 sections in the PE file, there will be exactly 5 members in this structure array. Each member of the array is equivalent to the each directory entry in the root directory.

That's all about the physical layout of the PE file format. The major steps in loading a PE file into memory is described as follow:

- When the PE file is running, the PE loader examines the DOS MZ header for the offset of the PE header. If it is found, it would skip to the PE header.
- The PE loader checks if the PE header is valid. If so, it goes to the end of the PE header.
- Following the PE header is the section table. The PE header reads information about the sections and maps those sections into memory using file mapping. It also gives each section the attributes as specified in the section table.
- After the PE file is mapped into memory, the PE loader concerns itself with the logical parts of the PE file, such as the import table.

The above steps are oversimplification and are based on my own observation. There may be some inaccuracies but it should give you the clear picture of the process.

4.1.2 PE header

PE header of second part of PE files structure. PE header includes important information of ma . As shown in Figure 4.2 PE header consists of four parts, a MS-DOS section, PE header, a section table, images pages.

The PE header starts with two characters "PE" as know as the PE header file signature.
The file header contains seven fields shown in Figure 4.3. The number of the members in section is determined by *NumberOfSection* file in file header.
The following of the file header is the Optional header which consists of three parts. The first part is the COFF standard fields which contains the sizes of various parts of code and the *AddressOfEntryPoint* which indicates the location of the entry point of the application and, the location of the end of the Import Address Table(IAT).
In order to create a fast malware classification system, classification malware system based on PE header's meta-data.

4.2 Decision tree[5]

Decision tree is a algorithm generally used in machine learning technique. Decision tree is to create a model that predicts the value of a target variable based on several input variables. An example

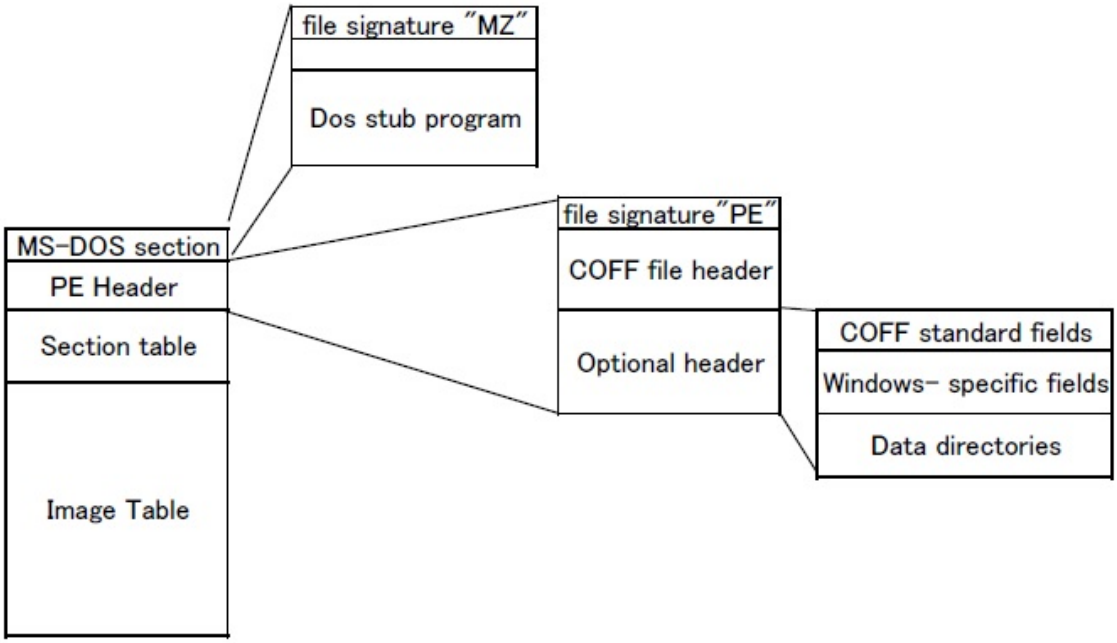

Figure 4.2: Layout a file in PE header format.

Member	Offset	Size
Machine	000000DC	Word
NumberOfSections	000000DE	Word
TimeDateStamp	000000E0	Dword
PointerToSymbolTable	000000E4	Dword
NumberOfSymbols	000000E8	Dword
SizeOfOptionalHeader	000000EC	Word
Characteristics	000000EE	Word

Figure 4.3: Layout of a file header

is shown on the right. Each interior node corresponds to one of the input variables; there are edges to children for each of the possible values of that input variable. Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf.

In data mining, trees can be described also as the combination of mathematical and computational techniques to aid the description, categorization and generalization of a given set of data.

Data comes in records of the form: $\mathbf{x}, Y = x_1, x_2, x_3, ..., x_k, Y$ The dependent variable, Y, is the target variable that we are trying to understand, classify or generalise. The vector x is composed of the input variables, x1, x2, x3 etc., that are used for that task.

4.3 Classification based on malware's meta-data using decision tree approach

To make malware classification rapid and correct, decision tree algorithm is used in malware classification system. As the previous section, Data comes in records of the form: $\mathbf{x}, Y = x_1, x_2, x_3, ..., x_k, Y$ In malware classification system, Y is malware families name, $x_1, x_2, x_3, ..., x_k, Y$ is meta-data. As we described in chapter 2, malware detected by anti-virus engines cannot used in this system. The new malware families is required for detect meaningful characterization of malware. This classification system use famous malware families that reported in Information technology agency such as netsky, mydoom, autorun, mytob, and other famous malware families. Therefore, when new malware is classified into famous malware families, researcher detect some meaningful characterize of malware, and reduce time for malware analysis.

Chapter 5

Implementation

5.1 Environment

5.2 Over view

Our system use machine learning technique, known as decision tree algorithm for malware classification system. Our goal is fast malware classification and we use decision tree algorithm to achieve that goal. Classification is a form of supervised learning, which requires training data, with known input/output, to form knowledge [33]. As shown in Figure 5.1, Our system contains three parts.

- First part: Read binary file to take meta data and input to database.
- Second part: Manually cluster malware which load from database.
- Third part: Use decision tree algorithm to classify malware.

We want to analyse a vast amount of malware so our system use database to store malware file's meta-data, to easily control a vast amount of data and easily share in the internet by web browser. The flow of data is shown in the Figure 5.1. We read binary file meta-data and input all of the meta-data in to database. In the second part, our system export meta-data in database as the training data to create the decision tree for classifying malware. Lastly, we use the decision tree

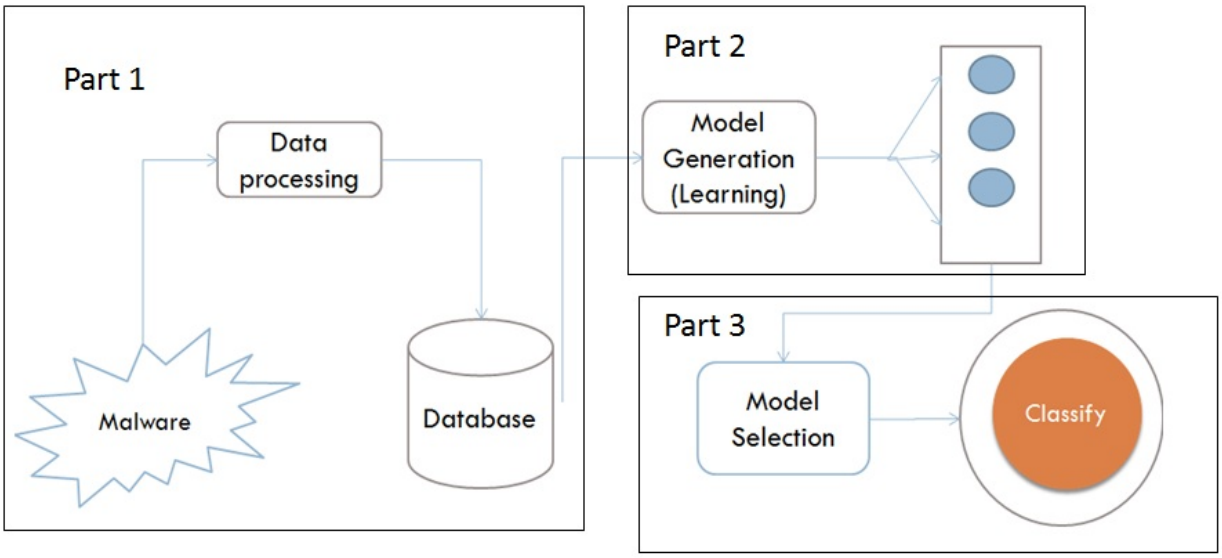

Figure 5.1: The system architecture.

algorithm which created in the second part to classify unknown malware into the different malware families.

5.3 Classification based on machine learning technique

5.3.1 Meta-data

In order to create a fast malware classification system, we only use PE header's meta-data to classify malware. PE header includes meta-data of MS-DOS section, COFF file header, Optional header, and Section header.

There is a problem that PE file's meta data value is so large that it is very difficult to detect unknown malware which have meta-data's different to the training data of our system. We index meta-data of PE header file which have semantic information for malware classification. For example, normally *ImageBase* field of PE header file have value 400000h [37] so that the value of ImageBase of malware file in our system is 0 if it has 40000h and it is 1 if it has other value, and in this research we call that value is semantic value. In this approach, all of PE file's meta-data is used. For example, consist of *HeaderFilesize*, *AddressOfEntryPoint*, *Sizeofsection*, *Imagebase*, *Numberofsection*, *StackCommit*, *SizeHeap Commit*, *SizeImage*, *Characteristics*. We calculated the semantic value of all malware file's meta-data.

5.3.2 Create training data

Virustotal is a service that analyzes malware and facilitates quick collection of viruses, worms, trojans, and all kinds of malware detected by antivirus vendor such as Norton and Kaspersky [**?**]. In this research, we use virustotal service to take the name of malware provided by Kaspersky and use it in order to cluster malware into malware families which have semantic information. We use http post technique to automatically get name from virus total service and cluster malware into malware families which shown in Figure **??**.

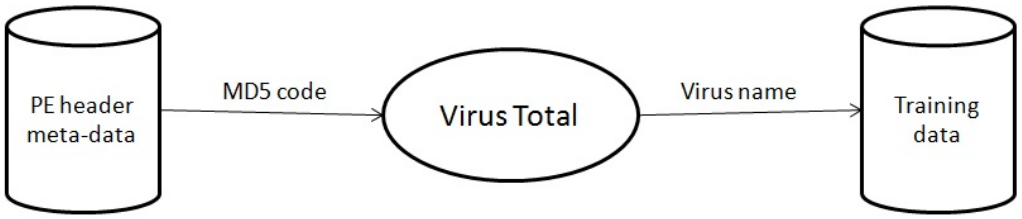

Figure 5.2: Clustering method.

5.3.3 Classification

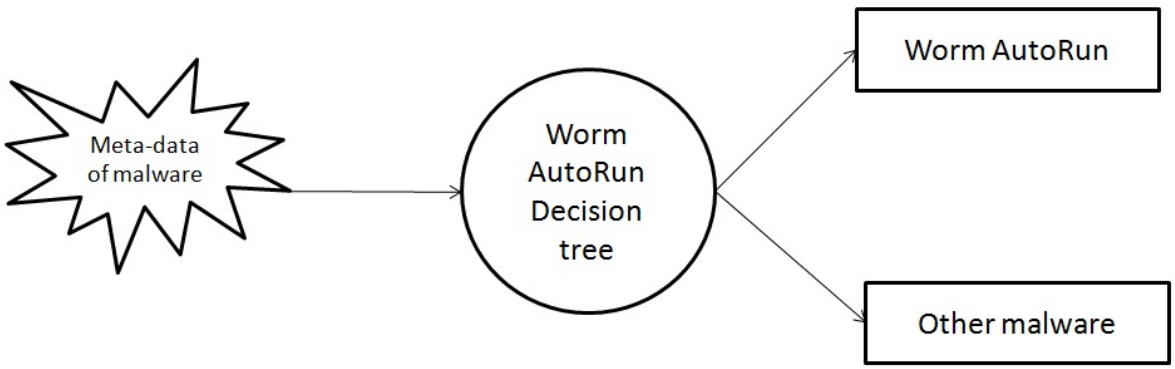

Figure 5.3: Worm autorun decision tree.

To make malware classification rapid and correct, we use decision tree algorithm. We make the

malware classification which easy to update a new malware family, our classification use decision tree algorithm to determine each malware family. For example, based on training data taken from clustering part, we create six decision trees. In the worm autorun decision tree, shown in Figure 5.3.

We use malware' meta-data as input data for each decision tree, and the the decision tree determine that the malware belongs to worm autorun family or not. If the input malware belong to worm autorun then malware classification detect the family that malware belong to, else we continue to the next decision tree. We use list malware PE header file's meta-data :Magic, MajorLinkerVersion, MinorLinkerVersion, SizeOfCode, SizeOfInitializedData, SizeOfUninitializedData, AddressOfEntryPoint, BaseOfCode, BaseOfData, ImageBase, SectionAlignment, FileAlignment, MajorOperatingSystemVersion, MinorOperatingSystemVersion, MajorImageVersion, MinorImageVersion, MajorSubsystemVersion, MinorSubsystemVersion, Reserved1, SizeOfImage, SizeOfHeaders, CheckSum, Subsystem, DllCharacteristics, SizeOfStackReserve, SizeOfStackCommit, SizeOfHeapReserve, SizeOfHeapCommit, LoaderFlags, NumberOfRvaAndSizes, Machine, NumberOfSections, TimeDateStamp, PointerToSymbolTable, NumberOfSymbols, SizeOfOptionalHeader, Characteristics. A value of each field in PE header have semantic for decision tree algorithm if it is appeared at 10% in malware training data. With this approach, we created the list semantic value of meta-data of malware. For example, this is list meta-data we created :1, 2, 6, 4, 0, 1, 1, 4, 0, 200, 4, 0, 0, 0, 4, 0, 0, 1.00E+00, 200, 0, 2, 0, 100000, 4000, 100000, 1000, 0, 10, 14c, 2, 1000, 8, 0, e0, 3, 818e. This is value before we calculated :35328, 10b, 2, 0, 6200, 0, 200, b32e, b000, 9000, 400000, 1000, 200, 3, 0, 0, 0, 4, 0, 0, d000, 400, afe8, 2, 0, 1000, 1000, 10000, 0, 0, 10, 14c, 6, 0, 0, 0, e0, 10e. Our malware classification part is shown in Figure 5.4. A training data taken from the malware clustering part is used to create the order of decision tree that make malware classification system correct. The decision tree we created, shown in Figure 5.5. PE header's meta-data of malware is inputted into the worn autorun decision tree in order to determine unknown malware. Then, the malware is determined belongs to worm auto run family or not.

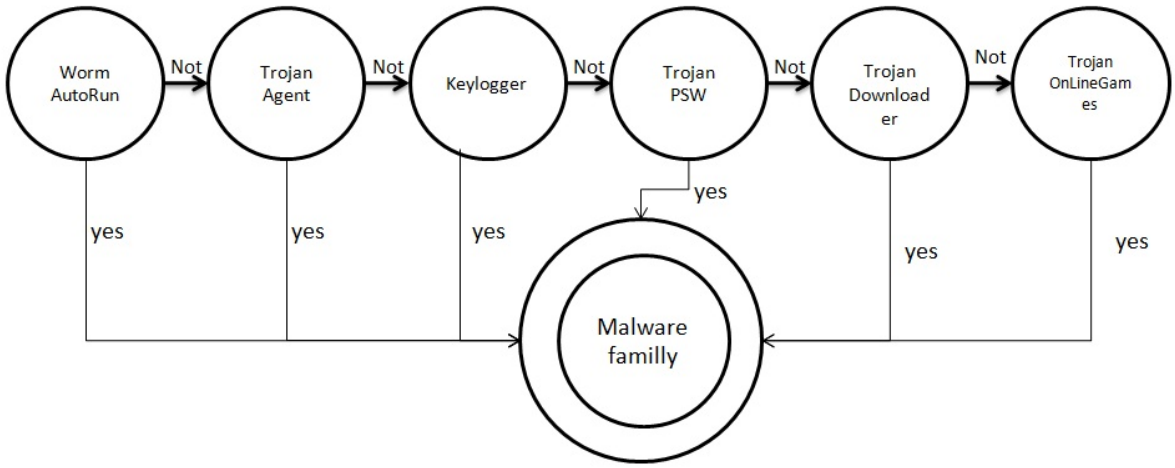

Figure 5.4: Malware classification system.

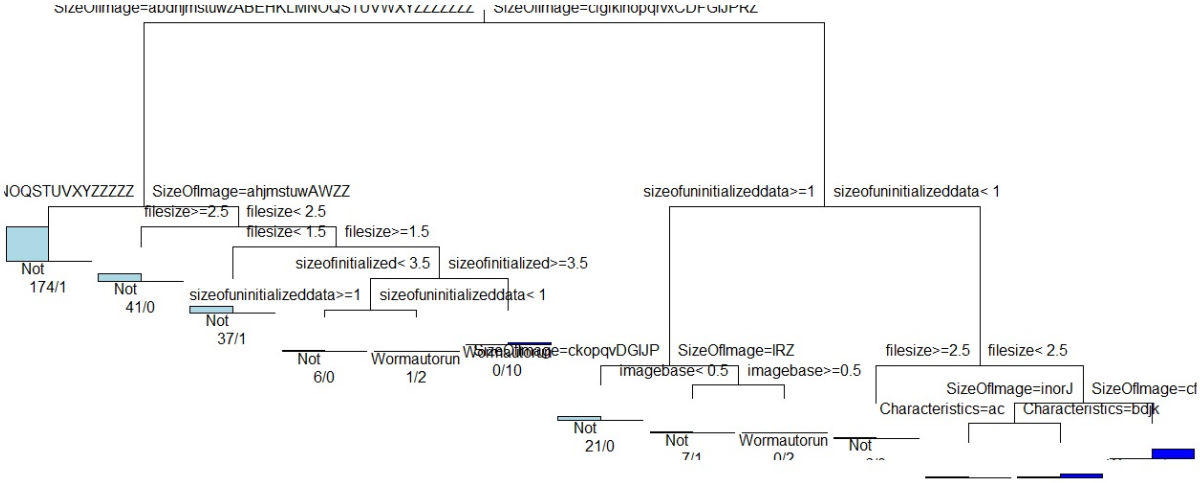

Figure 5.5: Worm autorun decision tree.

Chapter 6

Evaluation

6.1 Accuracy evaluation

As previously stated, we obtained ? malware PE file, used ? malware PE file for make 8 decision tree. Therefore, we used ? malware meta-data as training data, in order to sort 8 decision tree, and we make the best order of decision tree which shown in Figure 6.1, this order have the best experimental result with ? malware training data. Finally, malware meta-data to test experimental result our system. With experimental result, the order of decision tree is shown in Figure 6.1.

With the order of decision tree, we use 100 malware to test the experimental result our system. In Table 6.2, we show the experimental result of our system. Some of number in axis show these total number of malware in that family, and that number is separated into group that these malware has been classified by our system. For example, we have 4 malware in Win32/Virut family, and we successfully classify 3 malware into Win32/Virut family and have false to classify 1 malware into other family. Therefore, Our system also recognizes the trojan agent family with 75 % accuracy.

Our system is useful to help virus researcher determined the malware family that unknown malware belongs to. The malware family contains Win32/Virut, Win32/Autorun, Win32/IRCbot, Win32/Gaobot, Win32/Waledac, Win32/Downadup, Win32/Sality, W32.Mota, known as famous malware family. Virus research who know the family of malware can easily determined some semantic similarity between malware and showed their inner similarity in behavior and static malware characteristic.

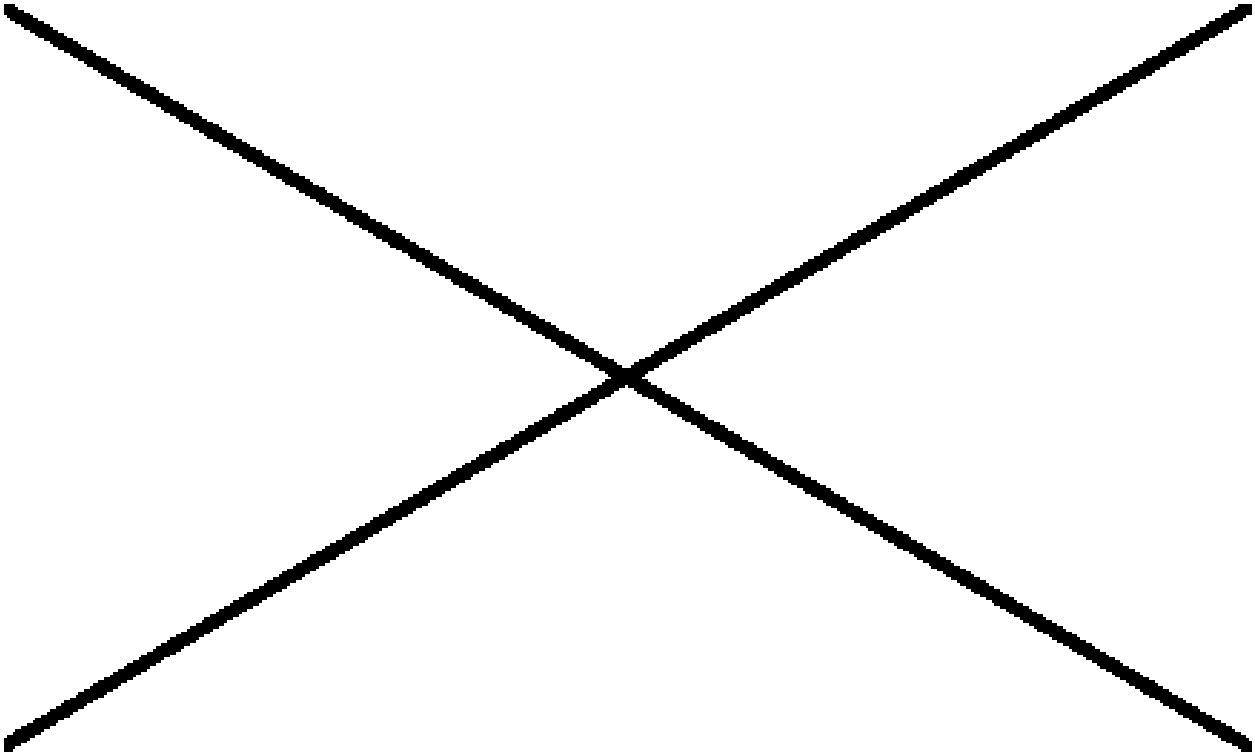

Figure 6.1: The best decision tree order.

Malware	Virut	Autorun	IRCbot	Gaobot	Waledac	Downadup	Sality	Mota	Accuracy
Virut	?	?	?	?	?	?	?	75%	
Autorun	0	0	0	?	?	?	?	50%	
IRCbot	0	0	2	0	1	0	0	66%	
Gaobot	1	0	0	2	8	0	3	53%	
Waledac	0	0	0	0	0	0	0	0%	
Downadup	0	0	0	0	0	1	1	50%	
Sality	0	0	0	0	0	1	1	50%	
Mota	4	4	2	3	15	2	41	57%	

Figure 6.2: Experimental result

6.2 Efficiency of classifying

Figure show comparing the step to detect semantic malware characterization using Virus total with our approach.

The figure 1 show classifying time in our system. The evaluation was performed on a 2.4 HZ

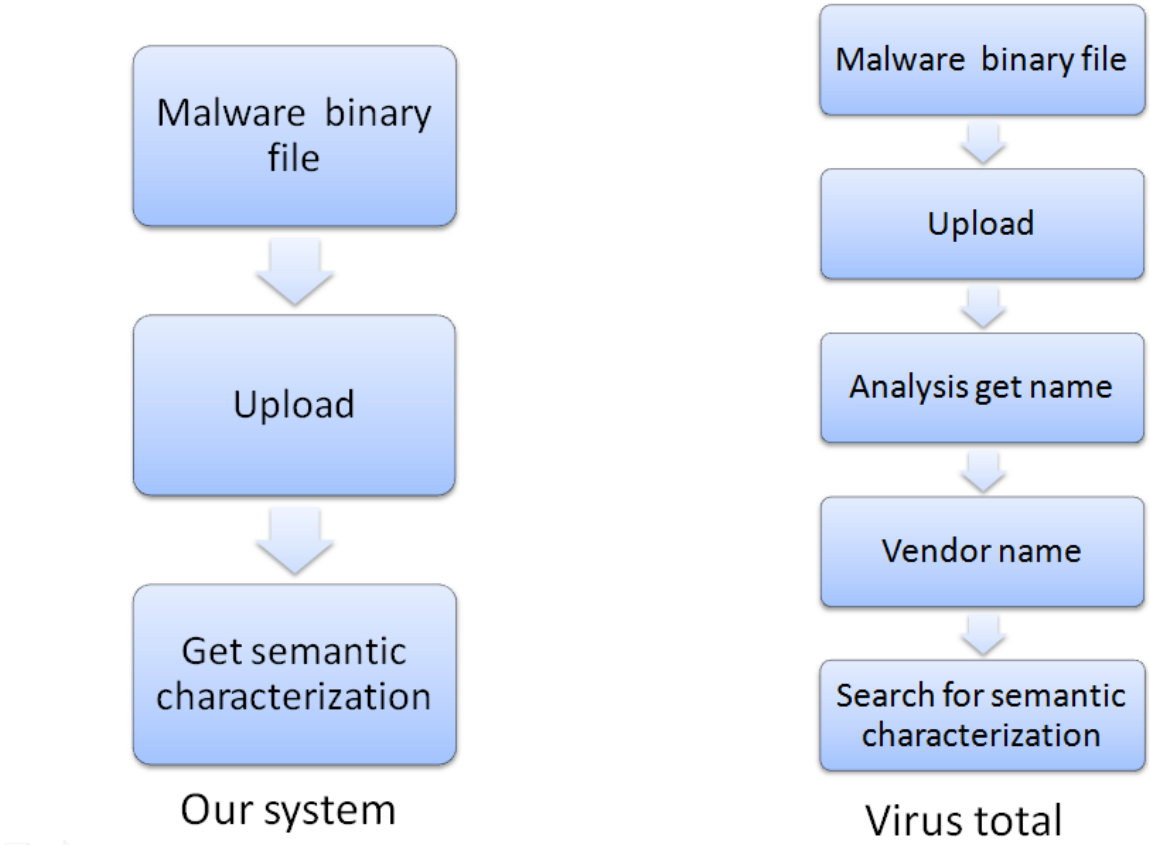

Figure 6.3: The best decision tree order.

core i3 laptop with 4G memory, running in ubuntu 11.4.

The result was shown in figure **??**. The median time to perform classification was 0.25 seconds. The slowest sample classified required 5.12 seconds. Only 6 samples required more than 2 seconds. Processing time of followgraph approach was shown in figure 6.4.

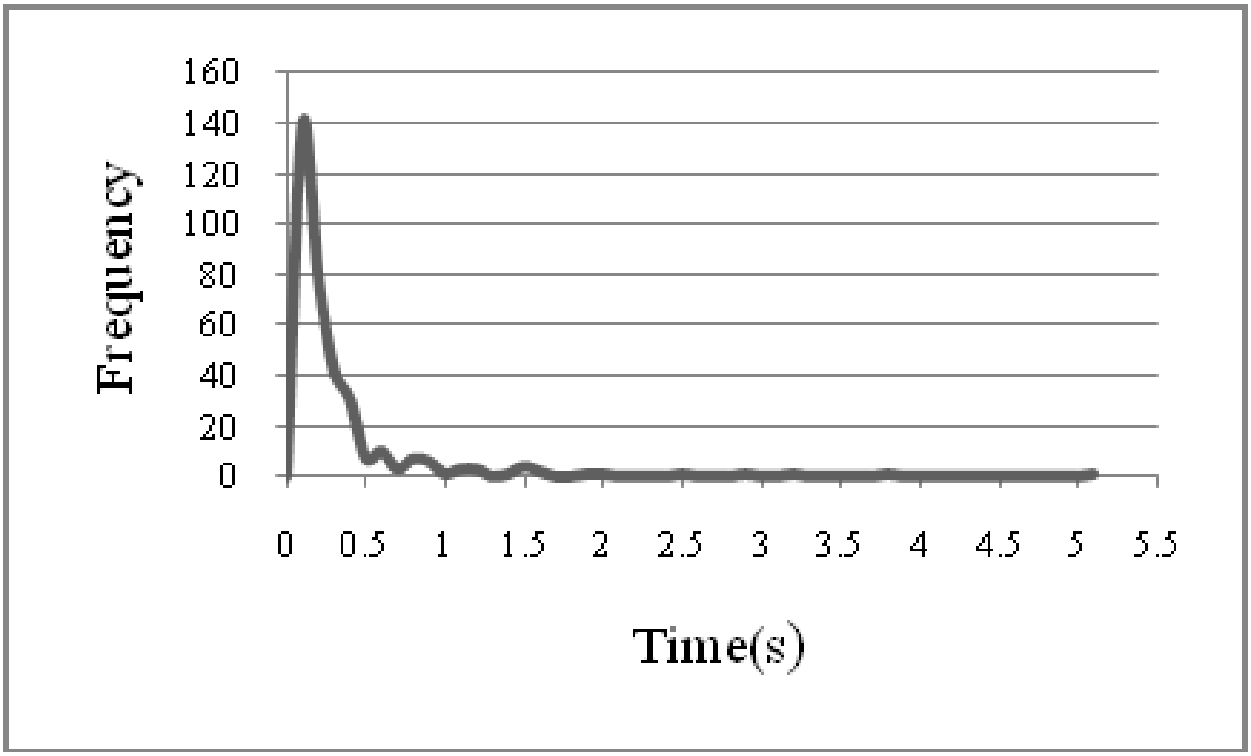

Figure 6.4: The best decision tree order.

Chapter 7

Conclusion

7.1 Conclusion

A vast amount of new malware sample each day is the difficult problem in malware analysis. Our malware fast malware classification system is successful to classify unknown malware into malware family which have semantic characteristic. We strongly believe that our system is useful for malware analysis to determine malware behavior and semantic malware characteristic to more easily analyze a vast amount of malware. However, there is a problem that our system use only malware meta-data and cannot detect the malware family which have same program structure.
Surprisingly, our system cannot be used to classify W32 malware which does not have the signature of PE file signature.

7.2 Future work

Our system successfully classified unknown malware into malware family. In the future work, we use Windows API in order to make our system determined unknown malware into the malware families, which have similarity of program structure.

Acknowledgement

First and foremost, I am sincerely grateful to faculty members in Murai and Tokuda Laboratory, Professor Hideyuki Tokuda, Professor Jun Murai, Associate Professor Hiroyuki Kusumoto, Professor Osamu Nakamura, Associate Professor Kazunori Takashio, Assistant Professor Noriyuki Shigechika, Assistant Professor Rodney D. Van Meter III, Associate Professor Keisuke Uehara, Associate Professor Jin Mitsugi, Lecturer Jin Nakazawa.

I would be extremely thankful to Professor Keiji Takeda for his endless help and valuable comments to my thesis. Without his help, I cannot even think of writing this thesis.

I own my special thank to Doctor Masayoshi Mizutani for his sincerely help and many advices since the time I first came to the lab.

I would like to thank to Master Kunihiko Shigematsu, Yuki Uehara, my friend Toshinori Usui, for listening and giving me many useful discussion points. Their gentleness and encouragement helped me overcome many hard times in this one year.

I would also like to thank all ISC group members, Sega-kun, Alc-kun, Lushe-kun, Nobita-kun, Fuyuki-kun, Yoshi-kun, Asp-chan, Hino-kun, Rocky-kun, Vigo-kun, who have given me numerous support and always cheer me up in my hard times.

I would like to thank to members in Murai and Tokuda Laboratory who have been so friendly and kindly supporting me with my research.

Bibliography

[1] David Zimmer. *http://www.woodmann.com/collaborative/tools/index.php/SysAnalyzer*. 21 March 2011.

[2] Symantec Announces MessageLabs Intelligence. *http://www.symantec.com/about/news/release/article.jsp prid=20101207*. 12 Dec 2011.

[3] John Bambenek. *http://handlers.dshield.org/jbambenek/keylogger.html*. 13 Sep 2005.

[4] Malware protection center. *http://www.microsoft.com/security/portal/Threat/Encyclopedia/Glossary.aspx* 21 Dec 2011.

[5] Wikipedia. *http://en.wikipedia.org/wiki/Decision-tree-learning*. 21 Dec 2011.

[6] Wikipedia. *http://en.wikipedia.org/wiki/Antivirus-software*. 14 Jun 2011.

[7] Eugene Kaspersky. *http://www.kaspersky.co.in/news?id=207576357*. 14 Jun 2011.

[8] Information-technology Promotion Agency, Japan.*http://www.ipa.go.jp/security/txt/list.html*. 01 Jan 2012.

[9] Malware Protection Center.*http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?=* *Win322fGaobotsummarylink*. 01 Jan 2012.

[10] Malware Protection Center.*http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?n* *Win32%2FIRCBot#symptoms_link*. 01 Jan 2012.

[11] Malware Protection Center.*http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?N* *Win32%2fAutorun*. 01 Jan 2012.

[12] Malware Protection Center.*http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?N* *Win32%2fVirut*. 01 Jan 2012.

[13] Qinghua Zhang, Douglas S. Reeve.*MetaAware: Identifying Metamorphic Malware*. 23rd Annual Computer Security Applications Conference 2007.

[14] Mohd Najwadi Yusoff and Aman Jantan.*Optimizing Decision Tree in Malware Classification System by using Genetic Algorithm*. The Society of Digital Information and Wireless Communications, 2011.

[15] MalwareChet Hosmer, Chief Scientis.*Polymorphic & Metamorphic Malware*. Black Hat, 2008.

[16] S. B. Kotsiantis.*Supervised Machine Learning: A Review of Classification Techniques*. Informatica 31, 2007.

[17] Paul Craig.*PE Packers Used in Malicious Software*. Ruxcon, 2006.

[18] Panda Security Internacional.*http://www.pandasecurity.com*. 01 Jan 2012.

[19] Malware Protection Center.*http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?N*
Win32%2fWaledac. 01 Jan 2012.

[20] Malware Protection Center.*http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?N*
Win32%2fSality. 01 Jan 2012.

[21] Symantec.*http://www.symantec.com/security_response/writeup.jsp?*
docid=2004070412024899. 01 Jan 2012.

[22] *http://news.softpedia.com/news/Kaspersky-Report-161-Million-Network-Attacks-Blocked-232812.shtml*. 14 December 2011.

[23] *http://win32assembly.online.fr/pe-tut1.html*. 14 December 2011.

[24] Wikipedia. *http://en.wikipedia.org/wiki/MD5*. 14 December 2011.

[25] Perdram amini, Ero Carrera *Reverse engineering on windows: Application in malicious code analysis*. Masters Thesis,Central Queensland University, 2010.

[26] *http://www.csn.ul.ie/ caolan/pub/winresdump/*
winresdump/doc/pefile.html. 24 Aug 2011.

[27] *http://www.viruslistjp.com/analysis/?pubid=204792101*. 17 Feb 2010.

[28] Silvio CESARE, *Fast Automated Unpacking and Classificationof Malware*. Masters Thesis,Central Queensland University, 2010.

[29] Jingjing Yao, Qiang Hou, *Malicious executables classification based on behavioral factor analysis*. International Conference on e-Education, e-Business, e-Management and e-Learning, 2010.

[30] Yuhei Kawakoya, Makoto Iwamura, Mitsutaka Itoh, *Analyzing Malware with Stealth Debugger*. Information Processing Society of Japan, 2008.

[31] Kevin Coogan, Saumya Debray, Tasneem Kaochar, Gregg Townsend, *Automatic Static Unpacking of Malware Binaries*. 16th Working Conference on Reverse Engineering, 2009.

[32] Tony Abou-Assaleh, Nick Cercone, Vlado Ke?selj, Ray Sweidan , *N-gram-based Detection of New Malicious Code*. Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004 .

[33] Tony Lee, Jigar J. Mody, *Behavioral Classification*. Presented at the EICAR Conference, May 2006.

[34] Yanfang Ye, Dingding Wang, Tao Li, Dongyi Ye, *IMDS: Intelligent Malware Detection System*. Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, 2007.

[35] Robin Sharp, *An Introduction to Malware*. Spring 2009.

[36] Georg Wicherski, *peHash: A Novel Approach to Fast Malware Clustering*. December 7, 2008.

[37] goppit, *PORTABLE EXECUTABLE FILE FORMAT*. 2011.