2011 First International Workshop on Complexity and Data Mining

Behavior-based Malware Analysis and Detection

LIU Wu[1)], REN Ping[2)], LIU Ke[3)], DUAN Hai-xin[1)]
[1)] Network Research Center of Tsinghua University, 100084 Beijing, P. R. China
[2)] School of Economics & Management, Chongqing Normal University
[3)] Beijing Technology and Business University, 100084 Beijing, P. R. China
liuwu@ccert.edu.cn

***Abstract*—Malware, such as Trojan Horse, Worms and Spyware severely threatens Internet. We observed that although malware and its variants may vary a lot from content signatures, they share some behavior features at a higher level which are more precise in revealing the real intent of malware. This paper investigates the technique of malware behavior extraction, presents the formal Malware Behavior Feature (MBF) extraction method, and proposes the malicious behavior feature based malware detection algorithm. Finally we designed and implemented the MBF based malware detection system, and the experimental results show that it can detect newly appeared unknown malwares.**

Keywords Malware Analysis; Malware Detection; Malicious Behavior; Network Security

I. Introduction and Related Works

Malware has become a major security threats to computer and networks (both intranet and internet). Malware analysis and detection is an essential technology that extracts the runtime behavior of malware and supplies signatures to detection systems and provides evidence for recovery, cleanup and forensics. Malware analysis is the premise and first-step of malware detection, its target is to extract the feature of malware which is used to detect the malware. There are basically two types of malware analysis: static analysis and dynamic analysis.

Static Malware Analysis (SMA) is the basic malware analysis method during which there is no need to execute the malware. It includes the following basic SMA methods: Basic Information Analysis, PE File Structure Analysis and Control Flow Analysis etc. To avoid the malware analysis and detection, more and more malwares adopts measures such as: Shell Code[1], Polymorphism[2] and Metamorphism[3] which make the analysis and detection of malware very difficult.

Different from SMA, the Dynamic Malware Analysis (DMA) needs to actually execute the malware in run-time to avoid the disturbance of Shell Code, Polymorphism and Metamorphism. There are mainly two types of DMAs: Debug Execution (such as WindDbg[4], SoftICE[5] and OllyDbg[6]) and Run-time Execution (such as FileMon[7], RegMon[8], TcpView[9] and process Explorer[10]).

The target of Malware Detection is to judge the existence of malware in a file.

Recently, the traditional Signature-based Malware Detection (SMD)[11] method is the mainstream in malware detection, and it is also the simple and effective method to detect malware. The basic principle of SMD is: manually analyze the collected malware, and extract its signature which can be used to distinguish and detect some malware from normal files uniquely.

It can be seen that, in SMD, it must first collect the malware sample before the detection, so it is time hysteretic, and can not detect the rapidly propagating malware effectively.

The Behavior-based Malware Detection (BMD) method [12] is quite different from SMD, it utilize the behavior information of the malware during its execution as the detection basis, which can get rid of the reliance on the feature of malware file itself. Because the behavior of malware is more unique than that of malware feature, for example, a malware can produce a new variation with different feature through shell code, Polymorphism and Metamorphism, but its behavior is still the same as the original one. So BMD will not be affected by shell code, Polymorphism and Metamorphism, and therefore can detect new malware. In recent years, there have been a lot of research works in BMD[13].

II. Malicious Behavior feature

To get a better understanding of malicious behavior intension, we analyze the behavior data of 236 known malwares (including Virus, Trojan and Worm etc.) which can be seen in Fig. 1. First, we can see that more than half (67%) of malware will produce sub-processes during its execution. Second, some of malwares will release and load kernel driver files during its execution, and finally, malwares will present some ordinary malicious behaviors such as thread injection, self-destroy (after the execution of malware sample, it will delete itself), and set the auto-run etc. We call these behaviors with apparent malicious intensions as the Malicious Behavior Feature (MBF).

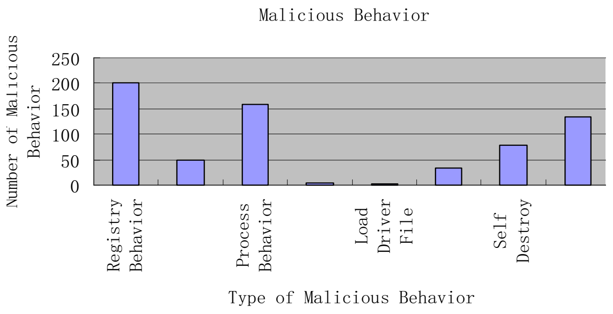


Figure 1 Statistics of known malware

This work is supported by grants from the National Natural Science Foundation of China (Grant No. 60203044, 90412010) , China 863 Project #2008BAH37B04 and Chongqing Science & technology Commission Program (CSTC2011AC2143)

III. EXTRACTION OF MALICIOUS BEHAVIOR FEATURE

In Windows operating systems, process is the basic execution unit and also the initiator of a behavior. To give a formal definition of the malicious behavior feature, we introduce concepts such as Behavior Operation (BO) and Behavior Operation Set (BOS).

A. Behavior Operation Set

Behavior Operation is the basic operations that leads to the change of system status by malware, for example, create a file, set a registry key etc. All the operations in malicious behavior events constitute the Behavior Operation Set (BOS).

According to different event types, we define 4 BOS:

(1) **File Actions (FA)**

FA={Create, Delete, Read, Write, Rename, LoadImage}

(2) Process Actions (PA)

PA={Create, Terninate}

(3) **Network Actions (NA)**

NA={TCP, UDP, IP} ⊕ {Listen, Connect, Send, Recv}

(4) **Registry Actions (RA)**

RA={OpenKey, CloseKey, CreateKey, DeleteKey, SetValue, DeleteValue, QueryValue}

The above 4 operations define all the operations used to extract the behavior data.

B. Entity Sets and Intruction Character of Action

Behavior Entity is the target object of Behavior Operation, and is the system level object, for example, File, Registry Key, and Network Port etc. The Entity Set is composed by all the behavior operations from the malware behavior data. According to different behavior and intension of malware, the Entity Set is divided into some Entity Sub-sets.

Let P_m denotes the malware process, the set P_C denotes the set of sub-process derived from the execution of malware, and the set MP denotes the set of derived from the malware process and all sub-processes derived from it, namely, $MP = \{P_m\} \cup P_C$.

Based on the above definitions, we can define the following BOS:

$$SF = \{Files\ affected\ by\ \mathrm{P} \mid \mathrm{P} \in \mathrm{MP}\}$$

$$SFC = \{Files\ created\ by\ \mathrm{P} \mid \mathrm{P} \in \mathrm{MP}\}$$

$$SFD = \{Files\ deleted\ by\ \mathrm{P} \mid \mathrm{P} \in \mathrm{MP}\}$$

$$SFW = \{Files\ written\ by\ \mathrm{P} \mid \mathrm{P} \in \mathrm{MP}\}$$

$$SNPL = \{LocalNetworkPorts\ opened\ by\ \mathrm{P} \mid \mathrm{P} \in \mathrm{MP}\}$$

$$SNPR = \{\mathrm{Re}\,moteNetworkPorts\ to\ which\ P\ attempts\ to\ connect \mid \mathrm{P} \in \mathrm{MP}\}$$

$$SNAL = \{LocalNetworkAddress\ used\ by\ \mathrm{P} \mid \mathrm{P} \in \mathrm{MP}\}$$

$$SNAR = \{\mathrm{Re}\,moteNetworkAddress\,used\ by\ \mathrm{P} \mid \mathrm{P} \in \mathrm{MP}\}$$

$$SRKC = \{\mathrm{Re}\,gistryKeys\ created\ by\ \mathrm{P} \mid \mathrm{P} \in \mathrm{MP}\}$$

$$SRKD = \{\mathrm{Re}\,gistryKeys\ deleted\ by\ \mathrm{P} \mid \mathrm{P} \in \mathrm{MP}\}$$

$$SRVS = \{\mathrm{Re}\,gistryKeyValues\ seted\ by\ \mathrm{P} \mid \mathrm{P} \in \mathrm{MP}\}$$

$$SRVD = \{\mathrm{Re}\,gistryKeyValues\ deleted\ by\ \mathrm{P} \mid \mathrm{P} \in \mathrm{MP}\}$$

For the convenience of detection, we define the following special BOS

$$SFS = \{System\,\mathrm{Pr}\,otectedalues\ Files\}$$

$$SFT = \{Trap\ Files\}$$

$$SFS = \{System\,\mathrm{Pr}\,otectedalues\ Files\}$$

$$SFM = \{Monitored\ Files\}$$

$$SFMD = \{Monitored\ Folders\}$$

$$SKM = \{Monitored\ \mathrm{Re}\,gistry\ Keys\}$$

We will describe the meanings and functions of the above BOS in the following sections.

Finally, we define the following operations to help define the malicious behavior feature:

(1) Addr(P): The network address bind to port P

(2) Value(RK): The value of registry key RK

(3) Action(F): The behavior operation acted on File F

(4) Image(P): The image file of process P

C. Formal Definition of Behavior Features

Malicious Behavior Feature is extracted from the analysis of the behavior data of known malwares and is used to detect the malware with similar behavior features. We define the Malicious Behavior Feature (MBF) as a triple:

<Feature_id, Mal_level, Bool_expression>

Where, Feature_id is a string identifier, which is used to uniquely mark the MBF; Mal_level is integer weight, which is used to mark the malicious degree of the MBF, and its values are: high, warning and low, where: high>warning>low; Bool_expression is a boolean expression defined on the Behavior Entity Set and the Behavior Operation Set, and is used to judge that whether a given malicious behavior data perform a specific MBF.

For example, the triple <Thread_Injection, high, $\exists \mathrm{F} \in \mathrm{SFC} \wedge \mathrm{P} \notin \mathrm{MP} \wedge \mathrm{P}\ Load\,\mathrm{Im}\,age\ F$ > defines a MBF with ID Thread_Injection, and its malicious degree is high. The meaning of the Boolean expression $\exists \mathrm{F} \in \mathrm{SFC} \wedge \mathrm{P} \notin \mathrm{MP} \wedge \mathrm{P}\ Load\,\mathrm{Im}\,age\ F$ is: if the malware released a file F during its execution, and in the meanwhile, there existed a process P automatically loaded the file F, and the process P was neither the malware itself nor derived from it. Then it denotes that this malware performed the thread injection behavior.

D. *Extraction of Malicious Behavior*

Based on the analysis of malicious behavior data, we successfully extracted some MBFs which can be seen in table 1.

Based on the MBFs in table 1, we will design the malicious behavior feature based detection system. But it needs to notice that, the MBFs in table 1 just the abstract description of the common malware intensions, they can not cover over all malicious behaviors.

Table 1 Extracted MBFs

No.	Feature_id	Mal_level	Expression
1	Thread_Injection	high	$\exists F \in SFC \wedge EP \notin MP \wedge P\ Load\mathrm{Im}age\ F$
2	Net_Scan	High	$\exists P \in SNPR \wedge Addr(P)\ is\ discret$
3	Net_Listen	Low	$SNPL \neq \Phi$
4	Net_Connect	Low	$\exists P \in SNPR \wedge Addr(P)\ \mathrm{is\ not}\ discret$
5	Self_Deletion	High	$\mathrm{Im}age(P_m) \in SFD$
6	Set_Autorun	Warning	
7	Extract_Exec	Warning	$\exists F \in SFC \wedge \exists P \in MP \wedge \mathrm{Im}age(P) = F$
8	Extract_Sys	Low	$\exists F.sys \in SFC$
9	Load_Sys	Low	$\exists F \in SFC \wedge \exists P \in MP \wedge P\ Load\mathrm{Im}age\ F.sys$
10	Write_System_Files	High	$\exists F \in SFS \wedge F \notin SFC \wedge Action(F) = Write$
11	Delete_System_Files	High	$(\exists F \in SFS \wedge F \notin SFC \wedge Action(F)$ $= \{Delete, \mathrm{Re}name\})$ $\vee(\exists F \in SFS \wedge F \in SFC \wedge Action(F) = Create)$
12	WX_Access_Trap_Files	High	$\exists F \in SF \wedge F \in SFT \wedge Action(F) \in$ $\{Write, Delete, \mathrm{Re}name, Create\}$
13	WX_Access_Trap_Folders	High	$\exists F \in SF \wedge Folder \in SFTD \wedge F\ In\ Folder$ $\wedge Action(F) \in \{Write, Delete, \mathrm{Re}name, Create\}$
14	WX_Access_Monitored_Folders	warning	$\exists F \in SF \wedge F \in SFM$ $\wedge Action(F) \in \{Write, Delete, \mathrm{Re}name, Create\}$
15	WX_Access_Monitored_Folders	warning	$\exists F \in SF \wedge Folder \in SFMD \wedge F\ In\ Folder$ $\wedge Action(F) \in \{Write, Delete, \mathrm{Re}name, Create\}$
16	WX_Access_Monitored_Regs	warning	$\exists RK \in SKM$ $\wedge Action(RK) \in \{Write, Delete, \mathrm{Re}name, Create\}$

IV. DETECTION ALGORITHM FOR MALICIOUS BEHAVIOR

Wih MBFs, we can now detect unknown malwares. The basic detection process is just to calculate the Boolean express value of the MBF with the malware behavior data. And the behavior based malware detection algorithm can be seen in Fig. 2. There are two inputs of the behavior based malware detection algorithm mw_detection: dr—malware behavior data; fea_list—malicious behavior features; The output dtr of mw_detection is the detection result and the judgment gist.

```
mw detection(DA RESULT dr, MAL FEATURE fea list)
{
    ba sets:=ContructBehaviorActionSets(dr);
    be sets:= ContructBehaviorEntitySets(dr);
    for (each feature Fea in fea list) {
        dtr:=EvaluateExpression(dr, ba sets, be sets, Fea.bool expression);
        if (SatisfiedMalFeature(dtr, Fea.feature id))
            AppendProof(dtr);
    }
}
```

Figure 2 Behavior based malware detection algorithm

First, both behavior operation set and behavior operation set are constructed from the collected malware behavior data. Second, for each MBF, calculate its Boolean expression and record the value V into the detection results dtr. The function SatisfiedMalFeature will judge the value V, if it is true, add the judgment gist by function AppendProof. The last function EvaluateMalLevel(dtr) in the algorithm is used to evaluate the malicious degree of the malware, if the malware shows more than one malicious behavior features, take the highest one.

V. DESIGN AND IMPLEMENTATION OF MALICIOUS BEHAVIOR DETECTION SYSTEM

Based on the above malware detection algorithm, we designed and implemented a MBF based malware detection system, which can be seen in Fig. 3.

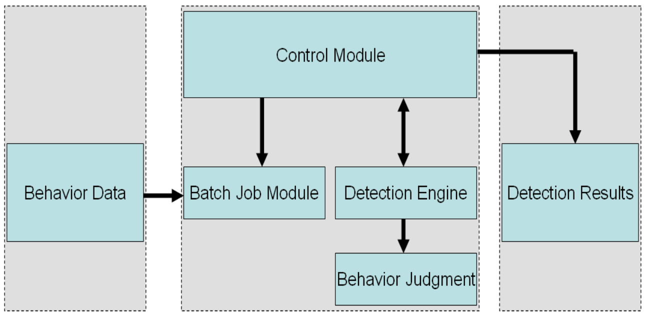


Figure 3 MBF based malware detection system

This detection system adopts the modular design. It takes the malware behavior data as the input and the output is the detection results of malware.

The Batch Job module is used to do the rapid batch detection for large number of malwares.

The Detection Engine module implements the behavior based detection algorithm, and it is implemented in the form of Dynamic Link Library in windows system to make it easy to update without affecting the main detection system.

The Behavior Judgment module is used to provide respective support functions for the Detection Engine.

Finally, the Control module outputs the detection result in text form. If the sample is judged as malware, the detection system will give the judgment gist.

VI. EXPERIMENTAL RESULTS

This section describes some experimental results of our behavior based malware detection system.

A. *False Alarm*

To test our detection system, we first test it with the 328 Win32 executable files in the system directory "C:\windows\". We ensure that all these files are not malwares. After the detection of our system, it shows that there are 7 of the 328 files (dplaysvr.exe, rtcshare.exe, nslookup.exe, tlntsvr.exe, ftp.exe, perfmon.exe and chkdsk.exe) are falsely judge as malwares, the false alarm is (7/328)*100%=2.13%. In other words, it means that the accuracy rate of our MBF based detection system is about 98%.

As the feature of some behavior type can not be as precise as that of malware sample, there exist inevitable false alarms for behavior based malware detection.

On the other hand, simply seen from the behavior, it seems that some normal programs do have a certain malicious natures. For example, after the execution of rtcshare.exe and tlntsvr.exe, they will be automatically listening local TCP ports 23 and 9492, which are similar to that of backdoor malwares.

Such false alarms can be eliminated through some technique methods. For example, when we find that the testing sample programs have signatures from Microsoft or other trusted third party, this sample should be normal programs other than malwares.

B. *Detection for New Malware*

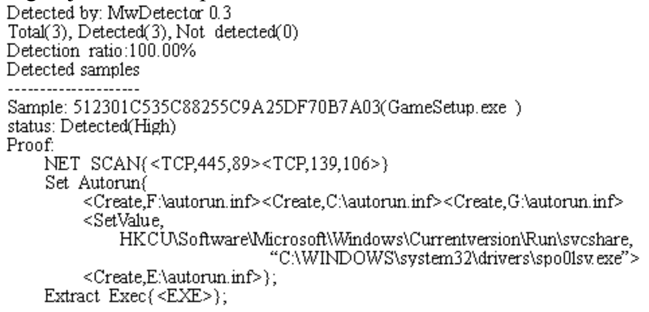
To test the ability to detect unknown malwares, we test both the Panda Burning Incense Virus sample and the Conficker worm sample with our detection system.

The detection results can be seen in table 2:

Table 2 Detection results for unknown malwares

Malware	Sample File Name	Detection Result	Matched MBFs
Panda Burning Incense	GameSetup.exe_	Detected (high)	NET_SCAN; elf_Deletion Set_Autorun; Extract_Exec WX_Access_Trap_Files WX_Access_Monitored_Folders
	Setup.exe_	Detected (high)	NET_SCAN; Set_Autorun Extract_Exec WX_Access_Trap_Files WX_Access_Monitored_Folders
	Sppoolsv.exe_	Detected (high)	NET_SCAN; Set_Autorun Extract_Exec WX_Access_Trap_Files WX_Access_Monitored_Folders
Conficker	LoadLib.exe	Detected (high)	Thread_Injection; NET_SCAN NET_LISTEN; NET_CONNECT WX_Access_ Monitored _Files WX_Access_Monitored_Folders WX_Access_Monitored_REGS

Table 2 shows that our detection system can detect newly out-broken unknown malwares. And in the meanwhile the detection system will output the detailed judgment gist, and Fig. 4 just shows a part of it.



```
Detected by: MwDetector 0.3
Total(3), Detected(3), Not  detected(0)
Detection  ratio:100.00%
Detected samples
--------------------
Sample: 512301C535C88255C9A25DF70B7A03(GameSetup.exe  )
status: Detected(High)
Proof:
     NET  SCAN{<TCP,445,89><TCP,139,106>}
     Set  Autorun{
          <Create,F:\autorun.inf><Create,C:\autorun.inf><Create,G:\autorun.inf>
          <SetValue,
               HKCU\Software\Microsoft\Windows\Currentversion\Run\svcshare,
                              "C:\WINDOWS\system32\drivers\spo0lsv.exe">
          <Create,E:\autorun.inf>};
     Extract  Exec{<EXE>};
```

Figure 4 The detailed judgment gist of the detected malware

CONCLUSION

In this paper, we investigate the technique of malware behavior extraction, present the formal Malware Behavior Feature (MBF) extraction method, and propose the malicious behavior feature based malware detection algorithm. Finally we designed and implemented the MBF based malware detection system, and the experimental results show that it can detect newly appeared unknown malwares.

REFERENCES

[1] Yang-seo, choi, Ik-kyun Kim, jin-tae Oh et al. PE File Header Analysis-based Packed PE File Detection Technique (PHAd). International Symposium on Computer Science and its Applications 2008.

[2] Polymorphic code. http://en.wikipedia.oarg/wiki/Polymorphic_code.

[3] Metamophic code. http:// /en.wikipedia.oarg/wiki/Metamorphic_code.

[4] WinDbg. http://www.microsoft.com/whdc/devtools/debugging/default.aspx

[5] SoftICE. http://en.wikipedia.org/wiki/SoftICE

[6] OllyDbg. http://www.ollydbg.de/

[7] FileMon. http://technet.microsoft.com/en-us/sysinternals/bb896642.aspx

[8] RegMon. http://technet.microsoft.com/en-us/sysinternals/bb896652.aspx

[9] TcpView.http://technet.microsoft.com/en-us/sysinternals/bb897437.aspx

[10] Process Explorer. http://technet.microsoft.com/en-us/sysinternals/bb896653.aspx.

[11] Sulaiman A, Ramamoorthy K, Mukkamala S et al. Disassembled code analyzer for malware. Information Reuse and Integration, Conf, 2005 IEEE International Conference on 2005, 398-403.

[12] Engin Kirda, Christopher Kruegel. Behavior-based Spyware Detection. The 15th conference on USENIX Security Symposium. Berkeley, CA, USA, 2006.

[13] Shih-Yao Dai, Sy-Yen Kuo. MAPMon: A Host-Based Malware Detection Tool. The 13th IEEE International Symposium on Pacific Rim Dependable Computing. 2007.