アンチウイルスソフトウエアのビヘイビアベースのマルウェア検知能力を評価する方法

陳 悦庭† 吉岡 克成‡ 松本 勉‡

†横浜国立大学
240-8501 横浜市保土ヶ谷区常盤台 79-1
f9190yuki@gmail.com, {yoshioka, tsutomu}@ynu.ac.jp

あらまし 本稿ではアンチウイルスソフトウエアのビヘイビアベースのマルウェア検知能力を評価するための手法を提案する．提案手法では，評価対象のアンチウイルスソフトウエアをインストールした動的解析環境と，インストールしていない環境を用意する．次に，それぞれの環境において，実マルウェア検体を実行して、アンチウイルスソフトウエアの存在がマルウェア検体の挙動に対し，どのような影響を与えるかを観察する．4つのアンチウイルスソフトウエアに対して提案手法を適用した結果，ビヘイビアベースの検知能力や検知時の対応に違いが確認された．

An Evaluation Method of Anti-virus Software on Capability of Behavior-based Malware Detection

Yueh-Ting Chen‡ Katsunari Yoshioka‡ Tsutomu Matsumoto‡

‡Yokohama National University
79-1 Tokiwadai, Hodokaya-ku, Yokohama 240-8501 Japan
f9190yuki@gmail.com, {yoshioka, tsutomu}@ynu.ac.jp

Abstract In this paper, we propose an explicit method to evaluate an anti-virus in terms of its behavior-based detection and protection capability. Namely, we construct two dynamic analysis environments, one with to-be-evaluated anti-virus installed and the other without it. Then, we execute various types of real malware samples on these environments and see what kind of behavioral changes the anti-virus would bring to the malware samples. We test our evaluation method with four anti-virus software products and find that their capability on behavior-based detection and their reaction upon the detection vary.

1 Introduction

In the modern society, computers play an important role in our daily lives. To avoid computers being compromised by malware, anti-virus software and security appliance are widely used. Consequently, malware authors have long been using obfuscation, polymorphism, and other techniques so that signature-based detection can be evaded. As the “arms race” between the attackers and security providers continues, recent security products claims to be equipped with behavior-based detection mechanism that does not rely on pre-defined signatures in order to follow up dynamically changing cyber-attacks.

Although there have been several efforts in evaluating anti-virus on their behavior-based malware detection, the details of the procedures, such as selection method of malware samples to be tested, and results of the evaluation experiments are not disclosed to public. In this paper, we propose an explicit method to evaluate an anti-virus in terms of its behavior-based detection and protection capability. Namely, we construct two dynamic analysis environments, one with the to-be-evaluated anti-virus installed and the other without it. Then, we execute various types of real malware samples on these environments and closely monitor their internal and network behaviors to see what kind of behavioral changes the anti-virus would bring to the malware

samples. We test our evaluation method with four anti-virus software and find that their capability on behavior-based detection and their reaction upon the detection vary, which may be of interest for their users.

2 Background

2.1 Existing Anti-virus Software Evaluation Methods

To evaluate quality of anti-virus software against malware protection, classical and real world methods, sometimes a combination of the two are used. In classical approach, anti-virus software is used to scan a folder containing a collection of currently active malicious programs or software. The percentage of false positives and false negatives is important parameter in classical testing.

Real world approach simulates the situations where malicious or infected websites or email attachments are opened on a protected system and calculates the proportion of threats that are detected and blocked.

2.2 Malware Analysis

There are two approaches on malware analysis: static analysis and dynamic analysis [2]. This chapter briefly introduces these two approaches and the tools for malware analysis.

2.2.1 Static Analysis

Static analysis also known as code analysis analyzes the file itself without execution. Eagle et al. describe that the detection patterns include string signature, byte-sequence n-grams, syntactic library call, control flow chart and operational code frequently distribution etc. [9].

The code has to be unpacked and decrypted before static analysis. Disassemble tools like IDA pro [10] and OllyDbg [11] would be helpful in static malware analysis, but memory dump tools like LordPE [12] and OllyDump [13] would also be a way to analyze malicious code by loading in the system memory and dumping it into a file when the packed code is unpacked and the original code is on the memory.

Memory dump also helps with code de-obfuscation. It is said that the limit of static analysis is to confront different type of tools and techniques to avoid detection of anti-virus software products. Christian et al. [8] describes detail on malware obfuscation technologies like dead code insertion, register reassessment, subroutine reordering, instruction substitution, code transposition and code integration to prevent from detecting by traditional defensive technologies like firewall, gateway, or even signature-based detection system.

2.2.2 Dynamic Analysis

The primary advantage of dynamic analysis is that it reveals the behavior of tested malware in black box manner. Dynamic Analysis observes malware behavior and analyzes its properties by executing malware samples in a testing environment such as sandbox, virtual machine, or even a physical machine.

Manuel E. [14] introduce couples of dynamic malware analysis techniques such as function call monitoring, function parameter analysis, information flow tracking, instruction trace, and auto-start extensibility points. They also introduce dynamic malware analysis tools like Anubis [15], CWSandbox [16], etc. Some dynamic analysis strategies [17] use multiple execution path exploration that would reveal malicious behavior with limited time or environment.

There are several tools available to conduct a dynamic analysis. We introduce Process Monitor, Regshot, and CaptureBat in the following sections.

2.3 Dynamic Analysis Tools

2.3.1 Process Monitor

Process Monitor [3] is an advanced monitoring tool for Windows that shows real-time file system, registry, and process/thread activity. There are many powerful monitoring and filtering capabilities allowing researchers capture the whole operation the system performs. However, there are many extra operations that are not observed by Process Monitor when analyzing the malware behavior.

One of the problems to use Process Monitor is that it is not an open-source software and we cannot modify the program to automate the dynamic analysis workflow.

2.3.2 Regshot

Regshot [4] is an open-source (LGPL) registry compare utility that allows quick comparison to the origin system state and the malware-modified one. It is an advantage that we can easily automate the dynamic analysis procedures using Regshot, but its functionality is not as powerful as Process Monitor. We could not get enough information from Regshot for us to analyze malware behavior.

2.3.3 CaptureBat

Christian et al. [1] introduce a behavioral analysis tool for the

Windows operating system family.

It is a powerful open-source tool that is able to monitor the state of a system during the execution of applications and processing the documents. It is installed in target system as a kernel driver recording the modification of the system. Figure-1 shows the CaptureBat structure [5].

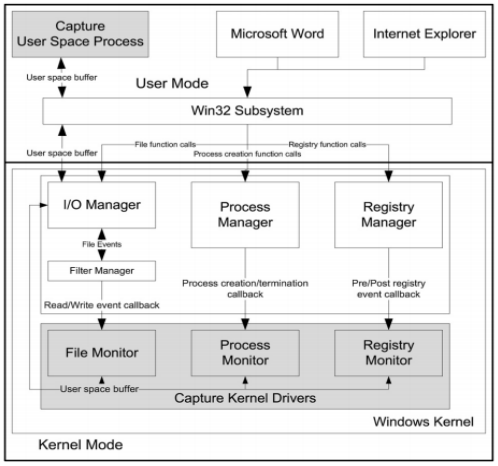

Figure-1 CaptureBat Structure in the system

3 Proposed Method

3.1 Method Overview

We propose a method to evaluate anti-virus software according to its behavioral based detection and protection capabilities.

The basic idea is that we check whether we can monitor malicious file writing, registry writing, process creating and suspicious connections that indicate the infection of the sample when malware sample is executed on an environment with anti-virus already installed.

A kind of file writing we focus on is modification of existing system files. We treat registry writing like file writing. Registry modification does not immediately mean that the system is infected. We analyze modified registry to make sure if it is malicious or not. Process creating is also a decisive activity to check the malware infection, but it depends on the activities of created processes. Suspicious connections are also checked to make sure the malware activities are ongoing.

We also investigate to which protection level antivirus software can protect system. We consider three protection levels such as protected, neutralized and compromised.

Protected means the malware could not perform its full malicious behavior when the anti-virus software is installed. Also the activities performed by malware would not cause a permanent damage to the anti-virus-software-installed system such as system file modification or deletion.

Neutralized, as well as protected category, means that the malware could not perform its full malicious behavior, but it would cause a permanent damage to the anti-virus software installed system.

Compromised means the malware sample runs successfully even though the anti-virus software is installed.

3.2 Evaluation Procedures

We construct two dynamic analysis environments as follow. Then, we execute various types of real malware samples on these environments and logs internal behaviors and network traffic for evaluation.

1. Environment without anti-virus installed
2. Environment with the to-be-evaluated anti-virus installed

In evaluating behavioral based detection capability of anti-virus software, we compare malware behaviors in environment 1 with environment 2.

We compare protection capabilities of different anti-virus software on environment 2. We find some anti-virus software include a signature-based malware removal tool, which examines an executable file every time it is about to be executed, stops the execution and remove the file if it is detected as malicious. In such case, we are interested in their behavior-based detection capability, we disable their signature-based removal tool in the experiment. The flow of our evaluation method is shown in Figure-2.

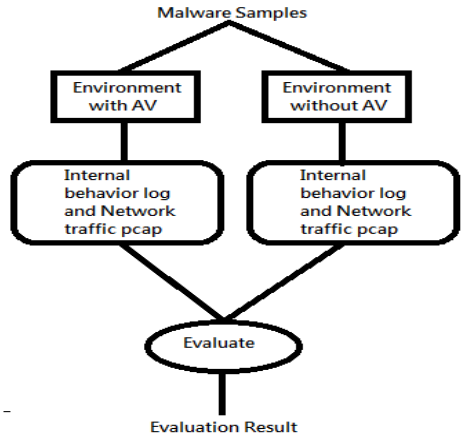

Figure-2 Flow of Evaluation Method

4 Experiment on the proposed method

The experiment is conducted during Nov. 2013 to Jan. 2014. We test our evaluation method with four anti-virus software products and then discuss if the obtained results would give insights on their capability of behavior-based detection. The malware used in this experiment is collected by low-interaction honeypot Dionaea [6] deployed at Yokohama National University. The counts of malware samples are 4,952.

Table.2 Malware Execution Result in Non-AV Environment

Total	4952
Not a Valid Win32 Application	951
After execution, Error Message Window displayed	463
Successful Execution	3538

4.1 Execution in Non-AV environment

We set up dynamic analysis environment with CaptureBat and Regshot on VirtualBox virtual machine as in Figure-3. We use Tcpdump for monitoring network traffic. Logs are collected in MySQL database.

We then execute 4,952 malware samples on it. We can execute 3,538 malware samples successfully. Some malware samples fail to be executed as they are not valid Win32 Application or due to lack of specific software malware samples utilize in test environment.

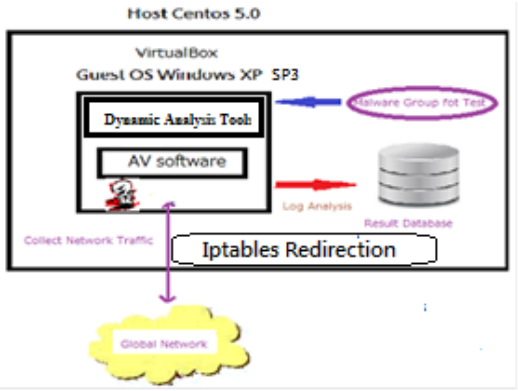

Figure-3 Analysis Environment

We also count behavior of malware samples in Table3. Most of the samples create new files such as executive files and batch files. There are also samples that create .com files to modify Internet configuration of the experiment system. Almost 90% of the malware samples write registry including creating a new one or modifying an existing one.

Table.3 Malware Behavior #Samples

Executable #Samples	3521
Registry Writing	3249
Executable File Creation	3141
Batch File Creation	589
DNS Network Traffic	2875

Detail explanation on registry activities is shown in Table.4, in which we find that the most frequently modified registry is Internet configuration related ones.

Table.4 Malware Behavior of Registry Writing

Modified Registry Key Name
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntrane
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyByPass
HKLM\SOFTWARE\Microsoft\Wireless\Client
HKLM\SOFTWARE\Microsoft\Wireless\ID
HKLM\Software\Microsoft\Windows\CurrentVersion\Run\Cryptographic Service
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Cache
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Cookies
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\path3\CachePath
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\path4\CachePath

As the malware samples used for this experiment are collected by Dionaea honeypot, most of them have network activities. Table 5 shows the most accessed domain names of DNS connections by malware samples. The list also contains benign domains as malware access these for various reasons [7].

Table.5 Request domain counts of distinct malware

Request Domain Name	#Samples
proxim.ntkrnlpa.info.	1198
Time.windows.com	1045
proxim.ircgalaxy.pl.	551
moscow-advokat.ru.	387
Scorti1.dns2go.com	117
m.drd3h.com.	92
home.najd.us.	91
ircd.zief.pl.	83
saber4.ircqforum.com.localdomain.	52
saber4.ircqforum.com.	52

4.2 Execution in AV-installed environment

We set up the same environment as in Figure 3 and install anti-virus on it. While executing malware sample, we assume that the users execute the malware by their will.

All of the anti-virus software is set up as default configuration, and all of them are updated to the version on June 25, 2014.

There are several samples fail to run in AV-installed environment even though they can run successfully in non-AV environment. This is because malware is blocked by DEP (Data Execution Prevention) [18] that is a Windows Security Defending System taken by anti-virus software or virtual machine detection techniques used by those samples to hide its malicious behavior for preventing from analyzing.

Several samples are deleted by anti-virus software automatically because of auto-removal tool for malware.

Comparative results of malware executions on Non-AV installed environment and AV-installed environment with each of AV-1, AV-2, AV-3 and AV-4 are shown in tables 6, 7, 8, and 9, respectively. In tables, **Successful Execution** means that the malware process is created successfully and performs some activities in our experiment system. Even though execution results belong to **Successful Execution** category, there can be a case when the system is not infected. **Runtime Error** means the malware process is created successfully, but the samples do not perform any malicious activities and showed the error message.

① AV-1 Result:

We find that most malware samples that can be run successfully in non-AV environment are not treated as valid executable format in AV-installed environment even though we do not change the system configuration between non-AV environment and AV-installed environment except AV-1 installed. It may be because of the protection mechanism like DEP that AV-1 implements. However we could not find any sign of dynamic detection to those samples that can be executed successfully in AV-1 installed environment.

Table.6 Result of AV-1

Non-AV environment Result	AV-Installed Environment Result	#Samples
Successful Execution	Not Executable Format	3490
	Successful Execution	56
RuntimeError	HardDiskError	1
	Not Executable Format	364
	RuntimeError	90
Not Executable Format	Not Executable Format	951

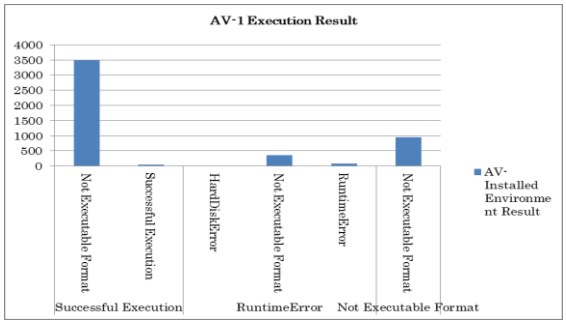

Figure-4 Result of AV-1

② AV-2 Result:

In case of AV-2, there are mostly two ways that executed samples are blocked: one is that these samples are treated as non-executable format like AV-1 and the other case is that the executed samples are removed by the auto-removal function of AV-2. Namely, the malware samples are deleted by AV-2 when the malware samples are copied to the file system. In fact, whenever the file system is being accessed, AV-2 would automatically scan the added files and delete them if they are detected as malicious. However, it is interesting that how the results change when the auto removal is switched off and only the behavior-based detection is active. We show the results in the last part of this chapter.

Table.7 Result of AV-2

Non-AV environment Result	AV-Installed Environment Result	#Samples
Successful Execution	Not Executable Format	2217
	RuntimeError	2
	Successful Execution	111
	Deleted by AV	1216
RuntimeError	Not Executable Format	111
	Deleted by AV	344
Not Executable Format	Not Executable Format	439
	Deleted by AV	512

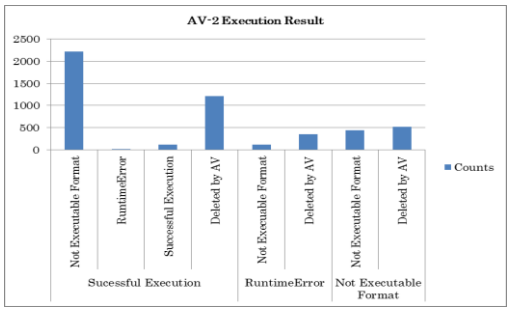

Figure-5 Result of AV-2

③ AV-3 Result:

There are 1,253 samples successfully executed even though AV-3 is installed. Many samples are also deleted before

execution by the auto-removal.

Table.8 Result of AV-3

Non-AV environment Result	AV-Installed Environment Result	#Samples
Successful Execution	Not Executable Format	6
	RuntimeError	5
	Successful Execution	1257
	Deleted by AV	2338
RuntimeError	Not Executable Format	91
	RuntimeError	115
	Deleted by AV	249
Not Executable Format	Not Executable Format	883
	RuntimeError	12
	Deleted by AV	56

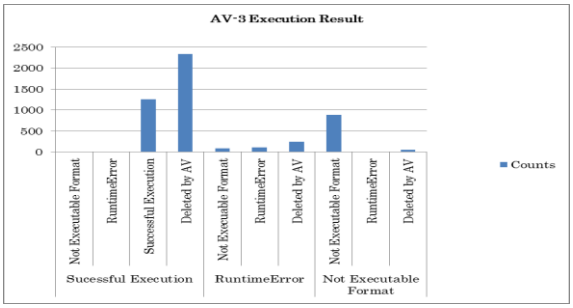

Figure-6 Result of AV-3

④ AV-4 Result:

AV-4 shows its excellent malware removal tool capability.

Table.9 Result of AV-4

Non-AV environment Result	AV-Installed Environment Result	#Samples
Successful Execution	Not Executable Format	4
	RuntimeError	3
	Successful Execution	407
	Deleted by AV	3132
RuntimeError	Not Executable Format	17
	RuntimeError	10
	Successful Execution	1
	Deleted by AV	427
Not Executable Format	Not Executable Format	650
	Deleted by AV	301

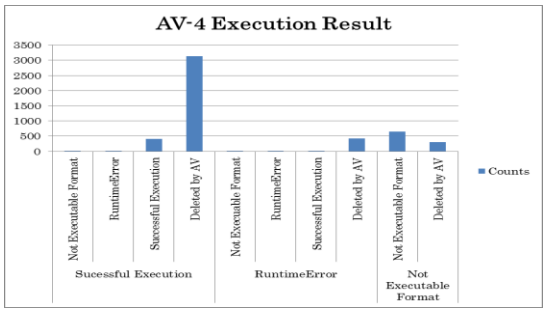

Figure-7 Result of AV-4

Table 10 and Figure-8 show the DNS connection counts that would be a reference for us to check the capability of blocking malicious connection. AV-1 counts the least. It is because of its protection approach that makes over 90% malware samples non-executable.

Table.10 AV's DNS connection

Snapshots Kinds	DNS Connection #Samples
Clean	2875
AV-1	234
Av-2	628
AV-3	1016
AV-4	1370

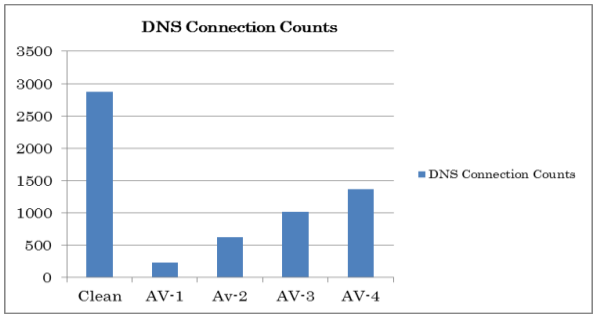

Figure-8 AV's DNS connection

4.3 Switching off Auto Removal Function

In the previous section, we notice that anti-virus often has auto-removal function that automatically checks and removes executable files before their execution. In experiment, in order to evaluate the solo capability of behavior-based detection, we declare the samples' directory in the exception list of such an auto-removal function. As AV-1 does not have the auto-removal capability, we would not take it into the extended experiments. In AV-3, exception list cannot be configured, so we would not take it into the experiments, too.

We take the log of anti-virus software as evidence that shows its detection capability. Table.11 and Table.12 show the result of AV-2 and AV-4.

We find lots of samples that create a new malicious executable file. When files created by those samples are deleted by anti-virus software, we denote such a case as **Protected**. The anti-virus software would detect malicious extended files creation and stop the malicious behavior. There are couples of samples intended to construct a malicious connection, but we observe that the anti-virus software would block those connection even though the executing malware sample is in the exception list.

Secondly, in **Neutralized** case, anti-virus software deletes the extended executable file created by the malware sample. Some samples change its behavior here. Those samples write the auto-start registry intending to drive the malicious

malware samples when the system starts up.

We observe that files created by some samples are deleted, but the samples continuously intend to take a malicious connection blocked by anti-virus software continuously. This causes the computer performance to slow down.

In **Compromised case**, samples would not be detected by anti-virus software. And it performs its full malicious behavior.

The following table shows the AV-2 execution result.

Table.11 Result of AV-2

AV-2 Deleted Samples		
Protected	**Neuralized**	**Compromised**
110	961	51

There are 1,216 samples that can be run in non-AV environment. When we switch off auto removal function 94 samples cannot be run in AV-installed environment because of DEP-like approach. We observe that there are lots of Neutralized cases in AV-2 result because of lots of registry modification by the samples. It would be difficult to differentiate that it is protected when the registry is being modified without the anti-virus software's re-modification.

Figure-9 shows that AV-2 can protect the system when the signature-based detection is enabled. However, Figure-10 shows that many samples are categorized as Netralized if we disable the signature-based detection approach.

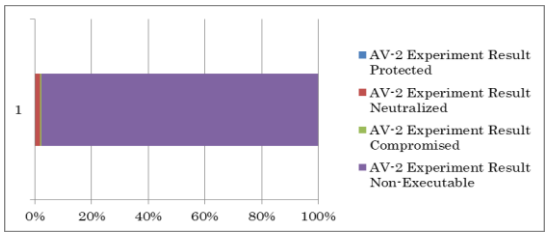

Figure-9 AV-2 Comparitive Result with Signature-based Detection

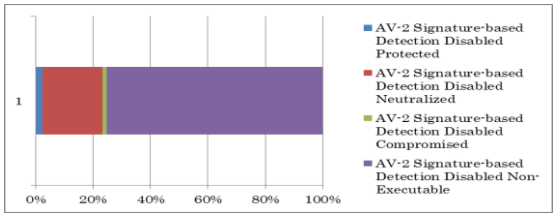

Figure-10 AV-2 Comparitive Result without Signature-based Detection

Table.12 Result of AV-4

AV-4 Deleted Samples		
Protected	**Neuralized**	**Compromised**
168	1568	62

According to table 12, we observe that number of Neutralized case is more than AV-2. There are over 1000 samples that cannot be run in AV-installed environment. It shows that it not only removes the malicious extended executable files but also do a job in processing the auto-restart registry and Internet configuration-related registry.

The comparative results of AV-4 are shown in Figure-11 and Figure-12.

When the anti-virus software loses its first-line protection ability of auto-removal tool, we observe that there are lots of malware samples that can be infected successfully. It would be the problem we want to emphasize.

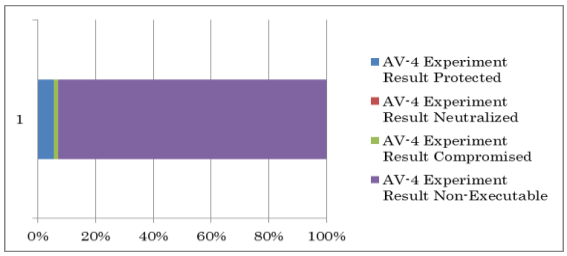

Figure-11 AV-4 Comparitive Result with Signature-based Detection

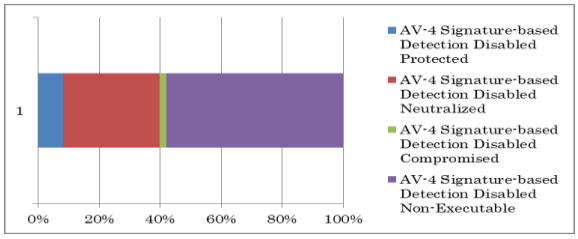

Figure-12 AV-4 Comparitive Result without Signature-based Detection

AV-2 and AV-4 show its detection capability to block the malicious activities even though the malware samples is in exception list and is not deleted by signature-based auto-removal tool. AV-2 is better than AV-4 with the ability to block the malicious connection.

We cannot disable the signature-based detection of AV-3, so we show the result of signature-based detection enabled with Figure-13. We can observe that almost 50% of malware samples are removed by AV-3.

Table.13 shows the result of those samples that cannot be

detected with signature-based detection. The result is interesting that there are not samples categorized into Neutralized.

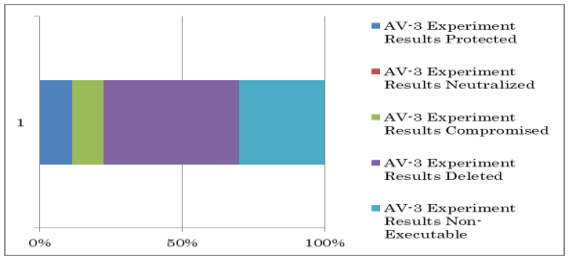

Figure-13 AV-3 Experiment Result with Signature-based Detection

Table.13 Result of AV-3

AV-3 Experiment Results		
Protected	**Neutralized**	**Compromised**
567	0	544

5 Conclusion

We propose an explicit method to evaluate an anti-virus in terms of its behavior-based detection and protection capabilities. With four anti-virus software and real malware samples, we find that our proposed method can highlight different capability of each anti-virus software products, which may be useful for their evaluation.

The problem of our experiment is that the environment is constructed with Virtual Box and thus anti-virtual-machine technology of malware may affect the results. This will be our future work to extend this study.

Acknowledgements

A part of this study has been supported by PRACTICE (Proactive Response Against Cyber-attacks Through International Collaborative Exchange) project by the Ministry of Internal Affairs and Communications, Japan.

References

[1] Honeypot Project, "Capture-BAT Download Page." http://www.honeynet.org/node/315 (Last Visit 2014/06/30)

[2] Ernst, Michael D, "Static and dynamic analysis: synergy and duality." Proceedings of the ACM-SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering. 2004.

[3] Russinovich, Mark, Bryce Cogswell, "Process Monitor v3.1." http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx (Last Visit 2014/06/30)

[4] regshot project, "regshot." http://regshot.sourceforge.net/ (Last Visit 2014/06/30)

[5] Seifert, Christian, et al, "Capture–A behavioral analysis tool for applications and documents." digital investigation 4 (2007): 23-30.

[6] dionaea, "dionaea catches bugs." http://dionaea.carnivore.it/ (Last Visit 2014/06/30)

[7] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. "Exposure: Finding malicious domains using passive dns analysis, " In NDSS. The Internet Society, 2011.

[8] Moser A, Kruegel C, Kirda E. "Limits of Static Analysis for Malware Detection," In 23rd Annual Computer Security Applications Conference (ACSAC); 2007.

[9] Egele, M., Scholte, T., Kirda, E., Kruegel, C. "A Survey on Automated Dynamic Malware Analysis Techniques and Tools," In ACM Computing Surveys 44 (2), 2012.

[10] 014 Hex-Rays SA, "IDAPro." https://www.hex-rays.com/products/ida/support/download_freeware.shtml (Last Visit 2014/06/30)

[11] 2000-2014 Oleh Yuschuk, "OllyDbg,." http://www.ollydbg.de/ (Last Visit 2014/06/30)

[12] Collaborative RCE Tool Library, "LordPE." http://www.woodmann.com/collaborative/tools/index.php/LordPE (Last Visit 2014/06/30)

[13] Collaborative RCE Tool Library, "OllyDump," http://www.woodmann.com/collaborative/tools/index.php/OllyDump (Last Visit 2014/06/30)

[14] Egele, Manuel, et al. "A survey on automated dynamic malware-analysis techniques and tools." ACM Computing Surveys (CSUR) 44.2 (2012): 6.

[15] International Secure Systems Lab, "Anubis. Analysis of unknown binaries." http://anubis.iseclab.org (Last Visit 2014/06/30)

[16] Willems, C., Holz, T., and Freiling, F, "Toward automated dynamic malware analysis using CWSandbox." IEEE Security and Privacy 5, 2, 32–39.

[17] Moser, Andreas, Christopher Kruegel, Engin Kirda. "Exploring multiple execution paths for malware analysis." Security and Privacy, 2007. SP'07. IEEE Symposium on. IEEE, 2007.

[18] TechNet Microsoft, " Data Execution Prevention. " http://technet.microsoft.com/en-us/library/cc738483(v=ws.10).aspx (Last Visit 2014/06/30)