Advanced Unknown Malicious Code Detection Model

Hyoungjun Kim[1], Jaehee Lee[2], Hyunsik Yoon[2], and Kyungho Lee[2]*
[1]Ahnlab, Seongnam-si, Republic of Korea
hyoungjun.kim@ahnlab.com
[2]Korea University, Seoul, Republic of Korea
{foodlook, hsyoon900827, kevinlee}@korea.ac.kr

Abstract

As IT technology increases, it became part of our life. Because of change in numbers, the files and data used in IT also increased. With simple data and file, malicious codes also increased in great number. This malicious code leads too many problems in the society. In fact, recently, new malicious codes that have not been detected yet are used in attacks such as APT (Advanced Persistent Threat). These codes became problem and lead to great damages. Thus, the speed of finding the undefined malicious code and making countermeasures became one of the important key words of the security. However, finding new malicious codes that is currently in action seems impossible. In order to find the malicious codes faster, there are researches on finding the special properties of the malicious code's file or action pattern. Through finding the properties of the codes, detecting the malicious codes became more effective and the results are satisfactory. This research will find the relation of malicious code's file property, behavior property, and property of each group or group as a hole in order to effectively detect the code that is suspicious of being malignant effectively and precisely. Thus, this paper will present a way to apply priority when it comes to detecting malicious code.

Keywords: Unknown Malicious Code Detection, Advanced Persistent Threat, Malicious Code Analysis

1 Introduction

As IT became part of our life and use of IT increases, files and data that is used in the area increased. Because of the data produced and used in IT increased greatly in number, technology of the malicious code and number of people creating malicious codes also increased in order to get in control of the data. According to one Security Company in Korea, three hundred thousand of malicious codes are gathered in a month. Thus, passively detecting the codes became nearly impossible. In order to differentiate malicious codes among the enormous amount of data in use, need of automation and efficiency brought about active research on the area. On the other hand, malicious codes also evolve as security technology increases. Many new technologies on malicious code are being generated in order to avoid the basic security system. Today, new malicious codes are in action consistently with out being detected by security system. To achieve it's malicious goals, codes use variety of mechanisms such as stealing the password information or creating traffic in order to attack other host by using connection of command control server [7]. In order to counteract to the malicious codes, there are researches on technology to detecting and analyzing the malicious codes. Anti-Malware Software's basically use Negative Security Model which is well known for Blacklist method. With Signature-based Pattern Matching False positives can be minimized. However, pattern matching methods have limits when it comes to finding new malicious

Research Briefs on Informaiton & Communication Technology Evolution (ReBICTE), Vol. 1, Article No. 9 (January 15, 2015)
*Corresponding author: Graduate School of Information Security, Korea University, Seoul, Republic of Korea, Tel: +82-2-3290-4885, Web: https://sites.google.com/site/kurmlab/

codes. To compensate the limits, Positive security, also called whitelist that allow only normal execution objects to be executed. However, this is limited to POS machines or production facilities which can define Whitelist.

Even if projects such as NSRL (National Software Reference Library) are in progress to commonly apply the technique, it is also limited only to certain countries or restricted environment. To compensate the limits on detecting malicious code in common host environment, there are researches going on to detect new and unknown malicious codes. Researches such as differentiating the malicious codes by applying behavior extraction algorithm in the process of the system function call in virtual environment system [10], Static detection of malicious code in executable programs [1], and malicious code detection by processing behavior-based detection with specific value on code's system call frequency[4] are efforts to detect the new and unknown malicious codes. These techniques focus on malicious codes technological function and property that tries to achieve its main function. With the idea, approaches are based on static or dynamic properties of the codes. However, if these detection techniques are used by themselves, it can be bypassed by methods such as code obfuscation.

Malicious code detection is a process of classifying normal object and abnormal object. Thus, there have been many researches to increase the accuracy of automated classification by using machine learning algorithm or data mining. Researches such as a malware classification method based on similarity of function structure [11], a malware classification method based on adaptive data compaction model using machine learning algorithm[12], a malware classification research by processing dynamic analysis using automated tools that traces its execution to group the samples that show similar behavior. With the analyses data, the generalized behavior profile is created to run by clustering algorithm[9], unknown malware detection using statistical analysis of byte-level file content [6], are classified as using machine learning algorithm or data mining. However, these researches need continuous improvement in accuracy since it has limit due to misclassification error which occurred by using specific data and certain classification model.

This paper classifies unknown malicious code's file and behavior characteristics that process host infection or malicious purpose to improve the preexisting researches that has limits sue to many reasons presented in the previous sentences. Based on each and groups characteristics relation, this paper will present a way to apply priority effectively when it comes to detecting malicious code.

2 Related Work

In this section, past researches on increasing the efficiency when detecting the malicious codes will be presented. After presenting the studies, this section will describe the limits of the past researches. At first, data mining methods for detection of new malicious is covered to explain how the data were gathered. Then, learning to detect malicious executables in the wild executables is explained. To increase the confidentiality, static analyses of executables to detect malicious pattern is processed. At last, data mining method for detection of new malicious executables is presented.

2.1 Data mining methods for detection of new malicious

In this paper[8], Schultz et al use several data mining techniques to distinguish between the benign and malicious executables in Windows or MSDOS format. They have done experiments on a dataset that consists of 1, 001 benign and 3, 265 malicious executables. These executables have 206 benign and 38 malicious samples in the portable executable (PE) file format. They have collected most of the benign executables from Windows 98 systems. They use three different approaches to statically extract features from executables. The first approach extracts DLL information inside PE executables. Further, the DLL

information is extracted using three types of feature vectors: (1) the list of DLLs (30 boolean values), (2) the list of DLL function calls (2, 229 boolean values), and (3) the number of different function calls within each DLL (30 integer values). RIPPER — an inductive rule-learning algorithm is used on top of every feature vector for classification. These schemes based on DLL information provide an overall detection accuracy of 83.62%, 88.36% and 89.07% respectively. Enough details about the DLLs are not provided, so we could not implement this scheme in our study. The second feature extraction approach extracts strings from the executables using GNU strings program. Native Bayes classifier is used on top of extracted strings for malware detection. This scheme provides an overall detection accuracy of 97.11%. This scheme is reported to give the best results amongst all, and we have implemented it for our comparative study. The third feature extraction approach uses byte sequences (ngrams) using hexdump. The authors do not explicitly specify the value of n used in their study. However, from an example provided in the paper, we deduce it to be 2 (bigrams). The MultiNative Bayes algorithm is used for classification. This algorithm uses voting by a collection of individual Native Bayes instances. This scheme provides an overall detection accuracy of 96.88%. The results of their experiments reveal that Native Bayes algorithm with strings is the most effective approach for detecting the unseen malicious executables with reasonable processing overheads. The authors acknowledge the fact that the string features are not robust and can be easily defeated. MultiNative Bayes with byte sequences also provides relatively high detection accuracy; however, it has large processing and memory requirements. Byte sequence technique was later improved by Kolter et al and is explained below.

2.2 Learning to detect malicious executables in the wild executables

Kolter et al use ngram analysis and data mining approaches to detect malicious executables in the wild.[5] They use ngram analysis to extract features from 1, 971 benign and 1, 651 malicious PE files. The PE files have been collected from machines running Windows 2000 and XP operating systems. The malicious PE files are taken from an older version of the VX Heavens Virus Collection. The authors evaluate their approach for two classification problems: (1) classification between the benign and malicious executables, and (2) categorization of executables as a function of their payload. The authors have categorized only three types — mailer, backdoor and virus — due to the limited number of malware samples. Top ngrams with the highest information gain are taken as binary features (T if present and F if absent) for every PE file. The authors have done pilot studies to determine the size of ngrams, the size of words and the number of top ngrams to be selected as features. A smaller dataset consisting of 561 benign and 476 malicious executables is considered in this study. They have used 4grams, one byte word and top 500 ngrams as features. Several inductive learning methods, namely instancebased learner, Native Bayes, support vector machines, decision trees and boosted versions of instance-based learner, Native Bayes, support vector machines and decision trees are used for classification. The same features are provided as input to all classifiers. They report the detecting accuracy as the area under an ROC curve (AUC) which is a more complete measure compared with the detection accuracy. AUCs show that the boosted decision trees outperform rest of the classifiers for both classification problems.
When the analysis techniques presented in the previous researches is used by itself, it can be avoided using Packing, code Obfuscation, or other bypassing technique. Thus, methodology to arrange and combine the techniques according to its detection efficiency is necessary. In next chapter, 26 rules that detect malicious codes behavior will be presented and described. Then, to place the rules efficiently, combined Detection Method which arranges the rules by analyzing the rules according to efficiency will be presented.

2.3 Static analysis of executables to detect malicious patterns

Mihai et al presents a unique viewpoint on malicious code detection.[2] He regard malicious code detection as an obfuscationdeobfuscation game between malicious code writers and researchers working on malicious code detection. Malicious code writers attempt to obfuscate the malicious code to subvert the malicious code detectors, such as antivirus software. They tested the resilience of three commercial virus scanners against codeobfuscation attacks. The results were surprising: the three commercial virus scanners could be subverted by very simple obfuscation transformations! They present an architecture for detecting malicious patterns in executables that is resilient to common obfuscation transformations. Experimental results demonstrate the efficacy of our prototype tool, SAFE (a static analyzer for executables).

2.4 Data Mining Methods for Detection of New Malicious Executables

Schultz et al presents serious security threat today is malicious executables, especially new, unseen malicious executables often arriving as email attachments.[3] These new malicious executables are created at the rate of thousands every year and pose a serious security threat. Current antivirus systems attempt to detect these new malicious programs with heuristics generated by hand. This approach is costly and oftentimes ineffective. We present a data mining framework that detects new, previously unseen malicious executables accurately and automatically. The data mining framework automatically found patterns in our data set and used these patterns to detect a set of new malicious binaries. Comparing our detection methods with a traditional signaturebased method, our method more than doubles the current detection rates for new malicious executables

3 Malicious code detection rule and grouping method

In order to detect unknown malicious code, detection technology and code behavior analysis is very important. Also, to distinguish the malicious code effectively, grouping of the detection rule by its efficiency is needed. Thus, this chapter will explain the 26 rules that were set to detect the the malicious code first. Then, hypothesis on effective arrangement of the rules will be suggested and tested its efficiency.

3.1 Malicious Code Detection Framework

This chapter will explain how and where each Rule detects malicious codes. Range can be majorly divided in to Disk and Memory. The Rule that research used is 26. Out of total Rule, 1$\tilde{1}$7 detect the malicious code in Sisk area. For 18$\tilde{2}$6, the test and detection of the malicious code is executed in Ram to detect the passive behavior of the code. The following Figure 1 demonstrates the theory. The 17 Rule that cover Disk area and 9 Rule that cover Ram area are grouped again with standard. The result of the grouping will be explained in detail in the next chapter.

3.2 Rule ID and Rule detection number set for malicious code detection

The rules that were set to detect the malicious code are 26. Each rule was referenced from Malware Analysts Cookbook and DVD 2010[11]'s recipe and rules classify the code through static or dynamic analysis. Rules are in following table. From the fact that well detecting Rule could detect normal code required another step in the process. The process used in the experiment is Focusing Group Interview. The professionals who participated in the Focusing Group Interview consist of personals who worked in the field of malicious code analysis for over 5 years. Through FGI, Rule was given assessment point

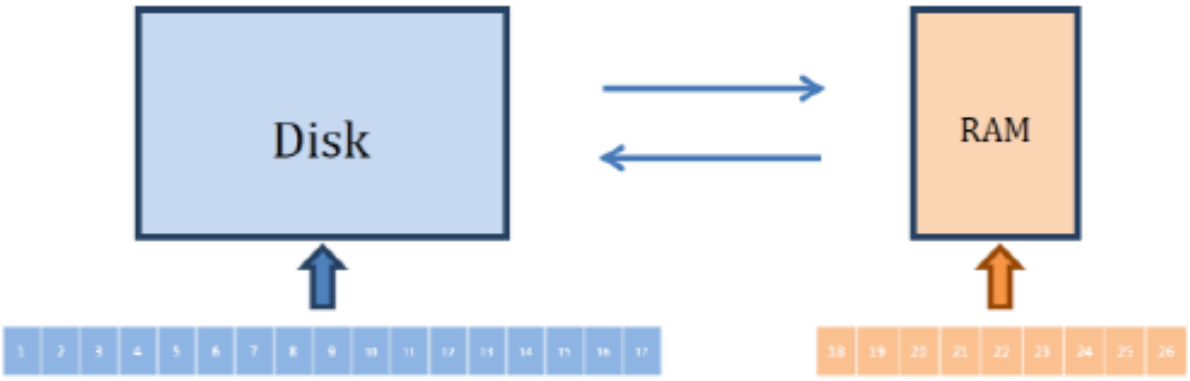

Figure 1: Malicious Code Detection Framework

to figure out the percentage of the Rule detecting the malicious code's behavior in the system. The calculation of Malicious Code Detection Rate is presented in the following Table 1.

Rule ID	Description	Detection Rate	Group Number	Group Description
1	Sections with extremely low or high entropy	29%	1	Entropy calculation
2	Abnormal section form detection	56%	2	Malicious/Suspicious Scan
3	Hidden/system/readonly files detection	17%		Malicious/Suspicious String Packer
4	Suspicious Section name	37%	3	Signature detection
5	Suspicious IAT entries (File search or control API)	2%		Process/Thread test
6	Suspicious IAT entries (INJECTION API)	10%		Patch/Hook detection
7	Packer Detection	46%	4	Stack/Heap test
8	Execution compaction (unknown packer) detection	37%		File generation/deletion/modification
9	Yara Script Detection	5%	5	Registry generation/deletion/modification
10	Process vulnerability attack attempt detection (HEAP SPRAY)	2%		Network connection
11	Hidden DLL	5%	6	Execution attempt
12	Hidden Thread	7%		Entropy calculation
13	Hidden Services	15%		Malicious/Suspicious Scan
14	Attempt of modifying memory of other process detection	7%	7	Malicious/Suspicious String
15	IAT hooking	2%		Packer
16	Shell code API calling attempt detection	2%	8	Signature detection

Rule ID	Description	DetectionRate	GroupNumber	Group Description
17	PE generation in system path detection	44%	9	Process/Thread test Patch/Hook detection Stack/Heap test
18	Compact executable file generation detection	54%		
19	Registry Create	68%	10	File generation/deletion/modification
20	Suspicious IP address connection attempt detection	5%	11	Registry generation/deletion /modification
21	Key input information interception attempt detection	10%	12	Network connection
22	System utility execution blocking	7%		
23	Execution in suspicious path detection	39%		
24	Batch file execution attempt detection	10%		

Table 1: Grouping Result

4 Efficiency measurement in malicious code detection rule

The total number of Unknown Malicious code samples is 37. These codes were randomly collected from malicious that were spread in commercial area from 2013 to 2014. Each sample was applied with the Rules used to detect malicious codes. Each sample is used to calculate the malicious code detection efficiency. This research concentrates on malicious code detection efficiency. Malicious code detection efficiency shows how accurately and rapidly the Rule can detect the malicious code. The reason is that in order to process the detection more effectively in thousands of files, Group that can detect malicious code faster has to be positioned in the front. This will allow detection time to be shortened.

4.1 Calculation of malicious code detection efficiency generated from detection rule combination

4.1.1 Combination of each rule according to detection efficiency

When the rules with high malicious code detection rate were send to front and top 2 15 rules were combined to apply on 41 samples to see how many malicious codes it can detect in Table 2.

Combination number	2	3	4~5	6~9	10~14	15
Detection number(n)	34	37	38	39	40	41
Detection rate(n/41)	82.9%	90.2%	92.7%	95.1%	97.6%	100%

Table 2: Detection efficiency of malicious code according to rule combination

4.1.2 Combination based on group detection efficiency

From the analyzed malicious code detection number per Rule, each groups average malicious code detection rate were calculated and apply higher weight on the group with higher malicious code detection number. Results are in Table 3.

Group Number	Description	FGI Score	Group Detection Rate
G1	Entropy calculation	7.0	32%
G2	Malicious/Suspicious Scan	4.0	84%
G3	Malicious/Suspicious String	3.3	51%
G4	Packer	3.5	62%
G5	Signature detection	7.0	8%
G6	Process/Thread test	6.0	30%
G7	Patch/Hook detection	6.0	8%
G8	Stack/Heap test	6.0	3%
G9	File generation/deletion/modification	1.0	92%
G10	Registry generation/deletion/modification	4.0	76%
G11	Network connection	10.0	5%
G12	Execution attempt	1.0	51%

Table 3: Group Weight result

According to the result of table 3, group with higher group weight showed high malicious code detection efficiency and can be applied in ordering of the Rule. This table result showed that the data used in the analysis were plausible. Through correlation analysis, a formula that can be used when actually arranging the malicious code Rules. The formula created for the detection is presented as following.

$$\mathbf{O}rderingscore = \mathbf{F}GIscore \times \mathbf{G}roupDetectionRate$$

However, Group Detection Rate has to be measured and calculate the Ordering Score whenever new Rule is added. This is very inconvenient and ineffective. So through regression analysis, calculating Ordering Score by measuring Group Detection Rate in the group is advisable. Regression analysis done with the result from correlation analysis is in table 4.

Model		Sum of Squares	df	MeanSquare	F	Sig.
1	Regression	.784	1	.784	23.834	.000[b]
	Residual	.724	22	.033		
	Total	1.507	23			
a. Subordination Variable: Group Detection Rate, b. Predictive value: FGI_Weight						

Table 4: Analysis of variance

With the result of the analysis, a Formula for Group Detection Rate can be presented as table 5.

If Ordering Score is added in the Formula for Group Detection Rate, the result will come out as presented in the Table 6.

If ordering score is decided according to score from the formula presented in the table 5 and table 6, FGI score is only thing that has to be calculated for Group Rule. Also, whenever Rule is added in the Group, FGI score will be renewed. When Ordering Score is measured with the techniques that used in

$\mathbf{Y} = -\mathbf{0.0}71X + \mathbf{C}$
Y : Group Detection Rate
X: FGI Score
C : Constant : 0.727

Table 5: The formula for Group Detection Rate

$\mathbf{O}rdering\ \mathbf{S}core = -\mathbf{0.0}71\mathbf{X^2} + \mathbf{0.7}27X$
Y : Group Detection Rate
X: FGI Score

Table 6: Applying Ordering Score in The formula for Group Detection Rate

the passage above, Group will be rearranged according to the score. The efficiency of the arrangement will be decided with actual detection rate of the actual malicious code. This result will allow people to get appropriate or proper arrangement so that detecting the malicious code can be most effective.

4.2 Advanced Unknown Malicious Code Detection Model

In order to detect unknown malicious codes, the malicious code detection Rules were combined and calculated the efficiency. Also, after putting the malicious code detection Rules in to groups, again, the efficiency was also calculated. In order to increase the efficiency of the malicious code detection, through FGI (Focus Group Interview), weight of each group were calculated. With the data that collected from the research until now, actual malicious code detection number and correlation were analyzed. With the result that extracted from the research, new ordering model that could be used in effectively detecting the unknown malicious codes and reporting the found malicious codes in the checking system is presented in the picture 2. And we also propose a Advanced Unknown Malicious Code Detection Model 3.

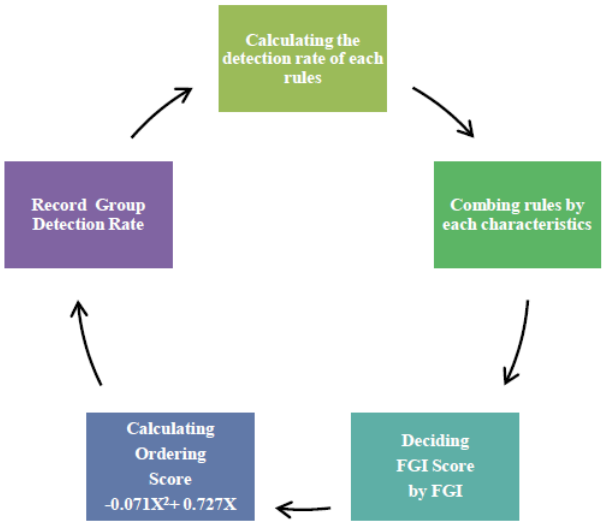

Figure 2: Malicious code detection Group Ordering Process

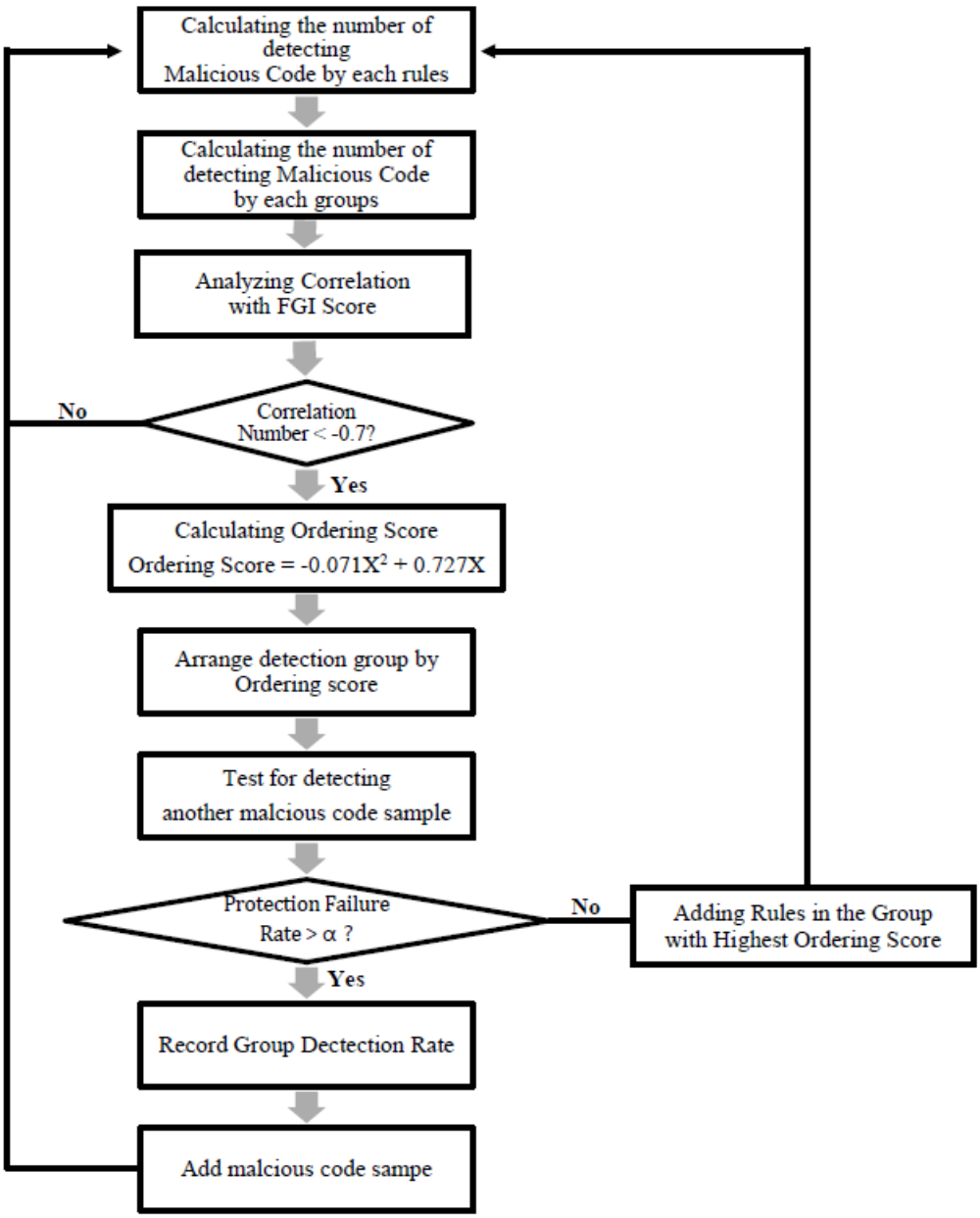

Figure 3: Advanced Unknown Malicious Code Detection Model

5 Conclusion

Today, use of malicious codes expanded for wider range. The examples are for monetary profit, governmental activity, or country to country cyber war. Because of the expansion of usage, number of the malicious code in internet or computers expanded greatly. Among created malicious codes, codes that have not detected for at list a year in activation or incubated for months are now starting to be detected. Because of the fact that malicious codes are dangerous to cyber world, importance of rapid detection of malicious code became a major issue in computer security. In this research, in order to classify unknown malicious codes rapidly, gathered malicious code sample were analyzed based on file and behavior's

characteristics. With analyzed data of characteristics, relation between groups was generated. Based on the result this research presented a way of increasing the malicious code detection by setting priority on the aspects that has to be applied. Also, Through Advanced Unknown Malicious Code Detection Model, in order to detect Unknown Malicious Code more rapidly, the Ordering Process of the group was suggested. However, verification on how fast and effectively the process has been improved is still a fact to be researched and analyzed. Through further research, the effectiveness of the process will be proved. As time goes by, changing IT environment, advancement of malicious code production technology and bypassing technique used on preexisting malware protection system allows endless change and transition of malicious code. As malicious codes change, its characteristics and techniques could also change. Thus, malicious characteristics criteria have to be updated based on the presented model by adjusting the priority to improve the usability. Also, to increase the accuracy of the model, additional analysis of other malicious code sample in greater number is important. Not only on malicious codes, is finding relation on normal files consideration on the further research. Moreover, research on detecting the malicious codes that are created by infecting the normal codes will be researched. Later, the way of decreasing the rate of finding miss detection will be calculated, analyzed and presented in the future.

References

[1] J. Bergeron, M. Debbabi, J. Desharnais, M. M. Erhioui, Y. Lavoie, and N. Tawbi. Static detection of malicious code in executable programs. In *Proc. of the 2001 Symposium on Requirements Engineering for Information Security (SREIS'01), Indianapolis, USA*, March 2001.

[2] M. Christodorescu and S. Jha. Static analysis of executables to detect malicious patterns. In *Proc of the 12th conference on USENIX Security Symposium (SSYM'03), Washington, DC, USA*, volume 12. USENIX Association, 2003.

[3] M. S. Eleazar, M. G. Schultz, E. Eskin, E. Zadok, and S. J. Stolfo. Data mining methods for detection of new malicious executables, 2007.

[4] S. jae Bae, J. ik Cho, T. shik Shon, and J. sub Moon. Malicious code detection using the effective preprocessing method based on native api. *Journal of the Korea Institute of Information Security and Cryptology*, 22(4):785–796, March 2012.

[5] J. Z. Kolter. Learning to detect malicious executables in the wild ABSTRACT, Apr. 01 2008.

[6] A. H. Michael Sikorski. *Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code*. William Pollack, 2012.

[7] A. H. Michael Sikorski. *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*. William Pollack, 2012.

[8] M. G. Schultz, E. Eskin, E. Zadok, and S. J. Stolfo. Data mining methods for detection of new malicious executables, Nov. 21 2010.

[9] M. Tabish, Z. Shafiq, and M. Farooq. A cooperative intrusion detection system for ad hoc networks. In *Proc. of the Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics (CSI-KDD'09), Fairfax, Virginia, USA*, pages 23–31. ACM, June 2009.

[10] G. Wagener, R. State, and A. Dulaunoy. Malware behaviour analysis. *Journal in Computer Virology*, 4(4):279–287, November 2008.

[11] Y. Zhong, H. Yamaki, and H. Takakura. A malware classification method based on similarity of function structure. In *Proc. of the 2012 IEEE/IPSJ 12th International Symposium on Applications and the Internet (SAIN'12), Izmir, Turkey*, pages 256–261. IEEE, July 2012.

[12] Y. Zhou and W. M. Inge. Malware detection using adaptive data compression. In D. Balfanz and J. Staddon, editors, *Proc. of the 1st ACM Workshop on Security and Artificial Intelligence AISec'08, Alexandria, VA, USA, October 27, 2008*, pages 53–60. ACM, 2008.

Author Biography

Hyounjun Kim is a Security Consultant and a Director of Security Consulting Bisuness Dision in Ahnlab and he is working on a Matster's Degree in the Department of Information Security at the Graduate school of Information Security, Korea University. His research interests are in the areas of malicious code detection analysis and ISMS.

Jaehee Leeis working on a Matster's Degree in the Department of Information Security at the Graduate school of Information Security, Korea University. His research interests are in the areas of risk management and quantum cryptology. He is researching SCADA Honeynet recently.

Hyunsik Yoon is working on a Matster's Degree in the Department of Information Security at the Graduate school of Information Security, Korea University. His research interests are in the areas of risk management, ISMS and consulting. He is researching IOT standard recently

Kyungho Lee is a accociate professor in graduate school of information security, Korea university. He received his B.S. degree in Mathematics from Sogang University. He received his M.S. degree in Information and communications from sogang university. He received is Ph.D in information security from Korea university, seoul, Korea.He was Vice president of STG security corporation in 1999,CEO of culsulting house corporation form 2002 to 2007, director of NHN corporation from 2007 to 2008, SEO of secubase corporation form 2008 to present time. He has been assistanct professor from 2011 to 2014 and associated professor form 2014 to present time. His research interests are in the areas of Risk Management, Information Security Consulting, Privacy Policy and Privacy Impact Assessment.