Network-based Executable File Extraction and Analysis for Malware Detection

Byoungkoo Kim[1,2], Ikkyun Kim[1] and Tai-Myoung Chung[2]
[1]*Network System Security Research Team, Electronics and Teletcommunicatons Research Institute, 161 Gajeong-dong, Yuseong-gu, Daejeon, 305-700, Republic of Korea*
[2]*Internet Management Technology Laboratory, Department of Electrical and Computer Engineering, Sungkyunkwan University, 300 Chenchen-dong, Jangan-gu, Suwon, Gyeonggi-do, 440-746, Republic of Korea*

Keywords: Network Packet, Malware Detection, Region Analysis, Executable File.

Abstract: The injury by various computer viruses is over the time comprised of the tendency to increase. Therefore, various methodologies for protecting the computer system from the threats of new malicious software are actively studied. In this paper, we present a network-based executable file extraction and analysis technique for malware detection. Here, an executable file extraction is processed by executable file specific session and pattern matching in reconfiguring hardware. Next, malware detection is processed by clustering analysis technique about an executable file which is divided into many regions. In other words, it detects a malware by measuring the byte distribution similarity between malicious executable files and normal executable files. The proposed technique can detect not only the known malicious software but also the unknown malicious software. Most of all, it uses network packets as analysis source unlike the existing host anti-virus techniques. Besides, the proposed detection technique easily can detect malicious software without complicated command analysis. Therefore, our approach can minimize the load on the system execution despite the load on the additional network packet processing.

1 INTRODUCTION

Most of the anti-virus software uses the file based diagnosis method. It takes only the segment or the intrinsic part of the executable file as the checking object. Therefore, it can minimize the un-detection and misdetection, and the fast scanning is possible. However, these methods can only correspond to the known malicious software. In order to overcome the limit, the heuristic detection method has been developed. These methods are classified into the mode of actually running on the virtual operating system and the mode of scanning the file itself without execution. That is, these are possible to performing the detection about unknown malicious software, but the information gathering about the commands within the actual file has to be preceded. So, it is easy that the system load on performing is caused. Therefore, while the efficient detection about unknown malicious software is carried out, the analysis technique minimizing the load of the accomplishment award is required. For resolving the problem, we propose the detection technique that it can detect not only the known malicious software but also unknown malicious software. It uses the network-based executable file extraction and analysis technique for malware detection. Most of all, the proposed technique easily can detect the malicious software without the complicated command analysis.

The remainder is structured as follows. The next section summarizes the work related to ours. Then, section 3 presents our network-based executable file extraction and analysis method for malware detection. Section 4 shows the experimental results. Finally, we conclude and suggest directions for further research in section 5.

2 RELATED WORK

This paper is mainly related to static anomaly-based detection. It uses the characteristics about the structure of packet or the program under inspection

Kim B., Kim I. and Chung T..
Network-based Executable File Extraction and Analysis for Malware Detection.
DOI: 10.5220/0004126104300433
In *Proceedings of the International Conference on Security and Cryptography* (SECRYPT-2012), pages 430-433
ISBN: 978-989-8565-24-2

for detecting malicious code. A key advantage of static anomaly based detection is that its use may make it possible to detect malware without having to allow the malware carrying program execute on the host system. In the previous works, several methods detect anomalies in the usage of network protocols by inspecting packet headers. The common denominator of them is the systematic application of learning techniques to automatically obtain profiles of normal behaviour for protocols at different layers. It uses time based models in which the probability of an event depends on the time since it last occurred, or proposes a computationally low cost approach to detecting anomalous traffic.

Kruegel et al. (2002) shows that it is possible to find the description of a system that computes a payload byte distribution and combines this information with extracted packet header features. In this approach, the resultant ASCII characters are sorted by frequency and then aggregated into six groups. Lee and Xiang (2001) used several information-theoretic measures, such as entropy and information gain, to evaluate the quality of anomaly detection methods, determine system parameters, and build models. These metrics help one to understand the fundamental properties of audit data. Wang and Stolfo (2005) present PAYL, a tool which calculates the expected payload for each service on a system. A byte frequency distribution is created which allows for a centroid model to be developed for each of the host services. Li et al. (2005) describe Fileprint (n-gram) analysis as a means for detecting malware. During the training phase, a model or set of models are derived that attempt to characterize the various file types on a system based on their structural composition. Anderson et al. (2004) proposed a search algorithm to detect the executable code transmitted in buffer overflow attacks. However, the algorithm only identified the operation of the buffer overflows attack by printing out the sequence of system calls used in the exploit. Besides, many studies have been carried out, such as Liu Wu et al. (2011) and Brijesh Kumar and Constantine Katsinis (2010).

3 THE PROPOSED METHOD

In this section, we briefly introduce the malware detection method by using the network-based executable file extraction and analysis technique. Here, we focus on Window PE (Portable Executable) file of executable file types. As shown in the figure 1, the proposed method consists of two part; H/W part for capturing network packets related to executable file extraction, and S/W part for malware detection. Through these techniques, our proposed method can perform the real-time malware detection as a network inline-mode.

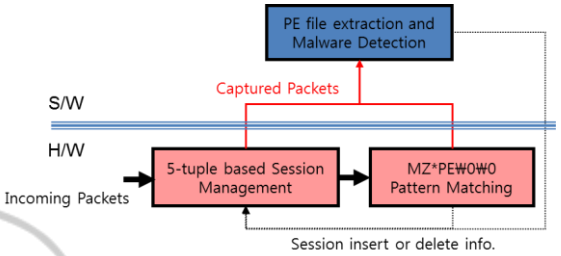

Figure 1: Network based malware detection architecture.

3.1 Executable File Extraction

An executable file extraction is processed by executable file specific session and pattern matching in reconfiguring hardware. As shown in the figure 2, Windows PE file starts from *"MZ(0x4D5A)"* pattern and includes *"PE00(0x50450000)"* pattern at the specified position.

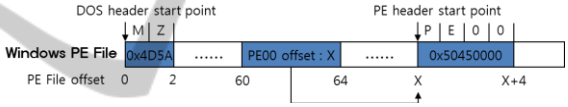

Figure 2: Windows PE file structure.

By using these characteristics, we can capture network packets related to Window PE file. Figure 3 shows a network packet capturing mechanism for extracting Windows PE files among a large quantity of network packets by using a hardware-based session tracking and pattern matching technology. That is, the method of extracting a Windows executable file includes: collecting incoming packets having a payload according to a session of a reference packet having an MZ pattern; performing a PE file pattern matching for the collected incoming packets; and forming a PE file based on at least one incoming packet satisfying the PE pattern matching.

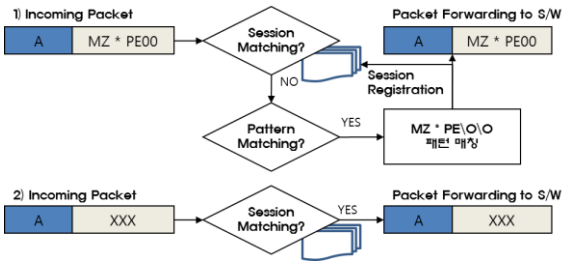

Figure 3: Network packet capturing mechanism.

3.2 Malware Detection Method

Malware detection is processed by clustering analysis technique about an executable file which is divided into multiple regions. In order that the proposed technique selects the clustering central value, it is performed having the population of each about normal executable files and malicious executable files. The information of these populations is used in inspecting each extracted executable file. Most of all, because the proposed technique is applied without the command analysis of the executable files, it is easily possible to detect a malicious executable file.

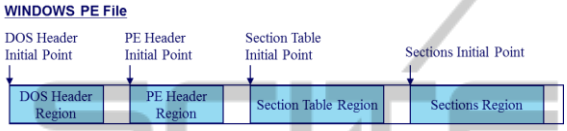

Figure 4: Region truncation of Windows PE file.

Figure 4 shows the conceptual diagram of region distribution about executable file. A format of Windows PE file can be divided into four regions according to the feature: DOS Header, PE Header, Section Table, and Sections. Here, the remaining part except for necessary information can be easily transformed by the malicious software manufacturer. That is, according to this division of territory, the region truncation about the Windows PE file can be performed. The initial point of each region becomes the initial point of each header of the Windows PE file structure. The initial points easily can be obtained through the parsing of the Windows PE file.

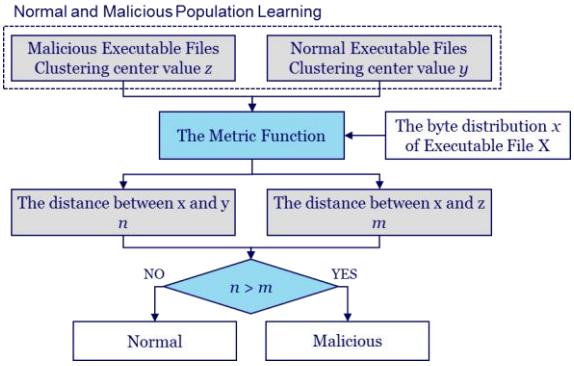

Figure 5: Malware detection mechanism.

Figure 5 shows the conceptual diagram of a malware detection mechanism. The population is distinguished from the normal executable files and malicious executable files. Then, the byte distribution value is calculated per the designated region of each population. Through the clustering learning about the calculated byte distribution values, each regional clustering central value is calculated. Then, the byte distribution similarity is measured between the extracted executable file and each population.

As shown in the figure, the byte distribution value *x* of the extracted executable file X is compared with the clustering central value *y* and *z* of each population. As a result of the metric function, the distance value *m* and *n* are calculated. Here, the distance value *n* indicates the distance between the clustering central value *y* of normal executable files and the byte distribution *x* of the extracted executable file X. If it is longer than the distance value *m* between the clustering central value *z* of malicious executable files and the byte distribution *x* of the extracted executable file X, it is determined as the malicious executable file. If not, it is normally judged. Through this operation, the proposed technique judges the malicious executable files. Here, we use the K-means algorithm as the clustering function and the Mahalanobis distance algorithm as the metric function.

4 EXPERIMENTAL RESULTS

We have developed network-based malware detection system based on our architecture, called ZASMIN (Kim et al., 2009). As shown in the figure 6, our system was implemented in a XILINX Virtex 4 platform FPGA. Also, the screen shots were captured during experiments to validate the performance of the prototype. In this experiment, we used a collection of malware executable files gathered from other external sources and normal Windows executable files under folder Sytem32 as a dataset.

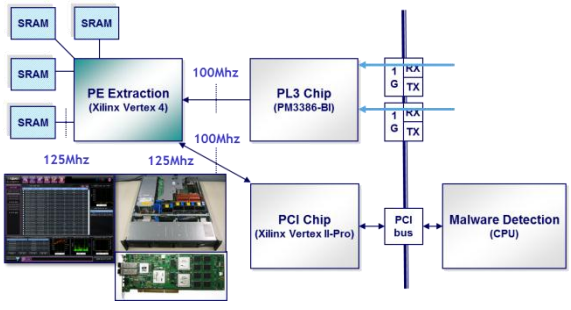

Figure 6: Implementation.

In this way, 1,850 Windows executable files and 845 virus executable files were used as training data

of each population. Then, our system monitored 256 normal executable files and 122 virus executable files transmitted for testing in test-bed network. As a result, all executable files have been extracted perfectly. Finally, the byte distribution values of the extracted executable files were compared with the clustering central values of each population. Here, the truncation size of each region is determined as the size which most well can distinguish between normal executable file and malicious executable file through learning tests of several times. Basically, the size of each region is more than the minimum 100 bytes for the data confidence. Most of all, we made a greater effort for minimizing the false positive rate, and maximizing the detection rate.

Table 1: Experimental results.

Exp.	T. size (byte)	Normal(256)		Malware(122)	
		False positives	C.	Detection rate	C.
A.	-	10%	26	93%	114
D.	120	1%	2	48%	59
P.	200	8%	21	73%	89
S.T.	160	1%	3	66%	81
S.	350	2%	5	44%	54

+ A.(All), D.(DOS header), P.(PE header), S.T.(Section Table), S.(Sections), C.(Count)

Our experimental result in this way is shown in the table 1. In the case of the extracted normal executable files, 230 executable files were altogether normally determined in each region. In the case of the other side, only 8 executable files were altogether normally determined in each region. That is, the normal executable files were normally judged with about 90% among 256 normal executable files for testing. On the other hand, the malicious executable files were as detected as 93% degree among 122 virus executable files for testing.

5 CONCLUSIONS

In this paper, we present the network-based executable file extraction and analysis technique for malware detection. The proposed technique can detect not only the known malicious software but also unknown malicious software. Most of all, our approach easily can detect the malicious software without the complicated command analysis. Therefore, it can minimize the load on the system execution. Besides, it can perform the real-time malware detection as a network inline-mode by using in reconfiguring hardware. Finally, we reported the experimental results of our approach. As shown in the experimental result, our approach showed a false positive rate under 10% and a detection rate over 90% beyond expectation. In future, we need to focus on reducing its false rate as the further study through more experimental results. Also, we will keep up our efforts for improvement in performance of detection mechanism on real world environment.

REFERENCES

Liu Wu, Ren Ping, Liu Ke, and Duan Hai-xin, 2011, 'Behavior-based Malware Analysis and Detection', In *Proceedings of the 2011 First International Workshop on Complexity and Data Mining*, Nanjing, China, pp. 39–42.

Brijesh Kumar and Constantine Katsinis, 2010, 'A Network Based Approach to Malware Detection in Large IT Infrastructures', In *Proceedings of the 2010 Ninth IEEE International Symposium on Network Computing and Applications*, MA, USA, pp. 188–191.

Ikkyun Kim, Daewon Kim, Byoungkoo Kim, Yangseo Choi, Seoungyong Yoon, Jintae Oh, and Jongsoo Jang, 2009. 'A case study of unknown attack detection against zero-day worm in the honeynet environment', In *Proceedings of the 11th international conference on Advanced Communication Technology*, NJ, USA, pp. 1715–1720.

Wei-Jen Li, Ke Wang, Salvatore J. Stolfo, and Benjamin Herzog, 2005. 'Fileprints: Identifying File Types by n-gram Analysis', In *Proceedings of the 2005 IEEE Workshop on Information Assurance and Security*, West Point, NY, USA, pp. 64–71.

Ke Wang, Gabriela Cretu, and Salvatore J. Stolfo, 2005. 'Anomalous Payload-based Worm Detection and Signature Generation', In *Symposium on Recent Advances in Intrusion Detection*, Seattle, WA, USA, pp. 227–246.

Stig Andersson, Andrew Clark, and George Mohay, 2004. 'Network-Based Buffer Overflow Detection by Exploit Code Analysis', In *Proceedings of the AusCERT Asia Pacific Information Technology Security Conference*, Gold Coast, Australia, pp. 23–27.

C. Kr¨ugel, T. Toth, and E. Kirda, 2002. 'Service Specific Anomaly Detection for Network Intrusion Detection', In *Proceedings of the 2002 ACM symposium on Applied computing*, NY, USA, pp. 201–208.

W. Lee and D. Xiang, 2001. 'Information-theoretic measures for anomaly detection', In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, Washington, DC, USA, pp. 130–143.