https://doi.org/10.7840/kics.2017.42.1.193

공격그룹 분류 및 예측을 위한 네트워크 행위기반 악성코드 분류에 관한 연구

임 효 영◆, 김 완 주*, 노 홍 준**, 임 재 성°

Research on Malware Classification with Network Activity for Classification and Attack Prediction of Attack Groups

Hyo-young Lim◆, Wan-ju Kim*, Hong-jun Noh**, Jae-sung Lim°

요 약

인터넷 시스템의 보안은 백신을 최신으로 업데이트하고, 신종 악성코드를 탐지해 내는 능력에 달려있다. 하지만, 급변하는 인터넷 환경과 더불어, 악성코드는 끊임없이 변종을 만들어내고 더욱 지능적으로 진화하고 있어 현재 운용중인 시그니쳐 기반 탐지체계로 탐지되지 않는다. 따라서, 본 연구에서는 악성코드의 네트워크 행위 패턴을 추출하여 DNA 서열 유사도를 비교하여 활용하는 유사 시퀀스 정렬 알고리즘을 적용하여 악성코드를 분류하는 기법을 제안한다. 제안한 기법을 실제 네트워크에서 수집된 악성코드 샘플 766개에 적용하여 유사도를 비교한 결과 40.4%의 정확도를 얻었다. 이는 코드나 다른 특성을 배제하고 악성코드의 네트워크 행위만으로 분류했다는 점을 미루어 볼 때 앞으로 더 발전 가능성이 있을 것으로 기대된다. 또한 이를 통해 공격그룹을 예측하거나 추가적인 공격을 예방할 수 있다.

Key Words : Malware Classification, Sequence Alignment, Clustering, Traffic Flow, Cyber Warfare

ABSTRACT

The security of Internet systems critically depends on the capability to keep anti-virus (AV) software up-to-date and maintain high detection accuracy against new malware. However, malware variants evolve so quickly they cannot be detected by conventional signature-based detection. In this paper, we proposed a malware classification method based on sequence patterns generated from the network flow of malware samples. We evaluated our method with 766 malware samples and obtained a classification accuracy of approximately 40.4%. In this study, malicious codes were classified only by network behavior of malicious codes, excluding codes and other characteristics. Therefore, this study is expected to be further developed in the future. Also, we can predict the attack groups and additional attacks can be prevented.

Ⅰ. Introduction

One of the major security threats on the Internet is malware, i.e., malicious software. According to the November 2014 McAfee Labs Threats Report[1], the total number of variants of malware in McAfee

※ 본 연구는 LIG넥스원의 지원으로 수행되었습니다.
◆ First Author : Ministry of National Defense, c15863@gmail.com, 정회원
° Corresponding Author : Ajou University Department of Computer Engineering, jaslim@ajou.ac.kr, 종신회원
* Ajou University Department of NCW, sizipus1@gmail.com, 정회원
** LIG넥스원 C4I연구소 통신연구센터, hongjum.noh@lignex1.com, 정회원
논문번호 : KICS2016-11-349, Received November 11, 2016; Revised December 5, 2016; Accepted December 6, 2016

Labs exceeded 200 million. Furthermore, malware has turned into a profitable business for malware authors and their customers. Malware authors often sell malware toolkits to their inexperienced customers, who can quickly create new customized malicious code variants.

The security of Internet systems critically depends on the capability to keep anti-virus (AV) software up-to-date and maintain high detection accuracy against new malware. However, malware variants evolve so quickly they cannot be detected by conventional signature-based detection. Furthermore, in contrast to the growing number of malicious codes, the number of analysts is markedly limited. Therefore, malware classification techniques have been proposed as solutions to deal with these problems.

Classification systems based on malware behavior are generally divided into two approaches. One relies on features extracted from the behavior of a system level, and the other depends on features extracted from network traffic. The vast majority of malware needs network activity in order to accomplish its purpose (e.g., downloading other malware, connecting to a C&C server, sending spam, stealing personal information, port scanning, and other typical network tasks). Many malware classification approaches based on network behavior have thus been proposed. Nevertheless, they merely rely on either a request URL or a payload for signature matching. To classify the network activity of malware, the patterns of network behavior must be understood and the changes in behavior observed. Therefore, flow sequences should be analyzed to provide interactive information of flow parameters caused by malware and their correlation.

In this paper, we present a novel malware classification method based on clustering of flow features and sequence alignment algorithms for computing sequence similarity, which represents the network behavior of malware. In our method, malware network behavior is translated into alphabetical sequence patterns. By modifying two sequence alignment algorithms, the behavior is analyzed to find the most similar malware. We focus on analyzing sequence similarity among the sequence patterns of malware traffic flow that is generated by executing malware on a dynamic analyzing system.

Ⅱ. Malware Classification

Malware classification is one of the main components of malware detection mechanisms. As stated by Rieck et al.[2], malware classification systems are necessary and important for detecting cyber threats because they work together with malware identification processes to produce correct and effective malware antidotes. Host-based anti-virus systems detect and remove malicious threats from end systems. As a normal part of this process, these anti-virus programs provide a description of the malware they detect. The ability of these products to successfully characterize these threats has far-reaching effects: from facilitating sharing across organizations to detecting the emergence of new threats.

However, for this information to be effective, the descriptions provided by these systems must be meaningful. The result in Bailey et al.[3] is striking in that there is a substantial difference in the number of unique labels created by each anti-virus system. While one might expect small differences, it is clear that anti-virus vendors disagree not only on what to label a piece of malware but also on how many unique labels exist for malware in general. Therefore, the technologies of malware classification must be developed effectively.

There are two major approaches to classifying malware: static and dynamic classification. The goal is to extract features and determine classifiers by which malware can be classified into its designated class. Malware classification generally consists of three steps. First, features that can distinguish each malware family are extracted. Second, a classifier is generated from the feature extracted. Finally, one identifies the family of malware using this classifier.

Cesare et al.[4] proposed a classification method of malware variants based on using similarity matching over sets of control flow graphs. They calculate the

similarity between malware programs using novel distance metrics of malware signatures. Kinable et al.[5] also presented control flow graphs for malware classification. In their method, clustering is related to classification and is used on call graphs using the graph edit distance to construct similarity matrices between malware samples. Control flow is observed to be more invariant in polymorphic malware.

Shankarapani et al.[6] presented a methodology for composing signatures of malicious codes from PEs for identifying known and unknown malware. The key assumption of their idea is that to preserve its functionality a polymorphic malware should contain a sufficiently similar API calling sequence or assembly code. Iwamoto et al.[7] proposed a malware classification that extracts features with API function calls. To visualize the grouping of samples with similar features they used hierarchical cluster analysis. Furthermore, Kim et al.[8] presented a the performance of malware detection system applying linear SVM machine learning classifier to detect Android malware application. Also, Kim et al.[9] proposed a android based malware detection mechanism using time-series analysis, which is one of statistical-based detection methods.

A large number of malware samples were dynamically analyzed using a sandbox environment and their behaviors collected. The behavior of malware can be divided into system and network behavior.

Rieck et al.[2] proposed a classification method of malware families based on their dynamic behavior. They adopted learning techniques to train the behavior features as classifiers and used them for classification decisions. Bayer et al.[10] proposed a clustering approach to identify and classify malware samples that exhibit similar behavior. They executed malware programs to compute distance using the LSH technique, which calculates a probabilistic estimate of all near pairs. Cho et al.[11] proposed a malware similarity method with malware executing. Based on sequence alignment, they computed the similarity of malware with API call sequences by removing repeated patterns.

Although most anti-virus software uses signature-matching techniques for detecting malware, Berger-Sabbatel et al.[12] revealed that this approach can be easily evaded. They presented a method for observing the communication patterns of executing malware with DNS replies that represent botnet activity. Stakhanova et al.[13] investigated malware behavior using network activity graphs. They modeled application level protocols as graph nodes and the commonalities between them as edges.

To focus on more specific information about network behavior, Perdisci et al.[14] addressed a malware clustering system by extracting HTTP traffic traces and analyzing their similarity. Differing from previous works, Rafique et al.[15] proposed a framework for extracting features from the protocol and traffic states in order to use the information obtained from all protocols. They assessed the applicability of various evolutionary and non-evolutionary algorithms for their malware family classification framework. Ahmed et al.[16] addressed the classification of packet contents to detect executable code in traffic.

Nari et al.[17] studied graph similarity analysis for classification of malware. Graphs are built with behavioral profiles that represent malware network activity such as DNS, SMTP, and HTTP. However, they did not consider the dependency on network flow or capture the network behavior of malware well enough to distinguish between different malware. In contrast to these approaches, we use only network traffic flow data and generate representations of malware network behavior for appropriate classification.

It is important not only for research on classification, but also for detecting malicious behavior in the network. Jung et al.[18] proposed a network defense mechanism based on isolated networks. In the paper, on their mechanism, every mobile device go through the integrity check system implemented in an isolated network, and can get the network access only if it has been validated successfully.

Sequence alignment is a method that compares two or more character sequences to obtain their similarities and dissimilarities. Malware

classification using sequence alignment has been extensively studied in malware analysis and detection research to classify normal, misuse, or unknown behavior. Several studies have proposed approaches to malware detection and classification. Inspired by the Smith-Waterman local alignment algorithm, Coull et al. presented a detection approach[19]. The authors later enhanced it and presented a sequence alignment method using binary scoring and a signature-updating scheme to detect masquerade attacks[20]. Another recent approach to detection is analyzing API call sequences and classifying them as benign or malicious[21].

Two techniques for malware classification using sequence alignment have recently been proposed[7,22]. Both extract more detailed information from binaries including sequences of API calls and graphical representations of control flow. We extend the previous studies that focused on network activity of sequencing features.

For similarity measurements between variants of malware, our research adopts the two sequence alignment algorithms: global alignment (Needleman-Wunsch) and local alignment (Smith-Waterman algorithms)[23,24]. These two algorithms belong to dynamic programming, which is a method for solving complex problems gradually using recurrence.

Ⅲ. Malware Classification Based on Network Pattern Using Sequence Alignment

In this chapter, we describe the proposed method for malware classification based on network behavior. Fig. 1. shows an outline of the proposed method, which is composed of training and classification phases.

3.1 Feature Extraction from Network Traffic of Malware

The goal of this work is to classify unknown malware in accordance with a sequence pattern of observable features.

The features are extracted from the network

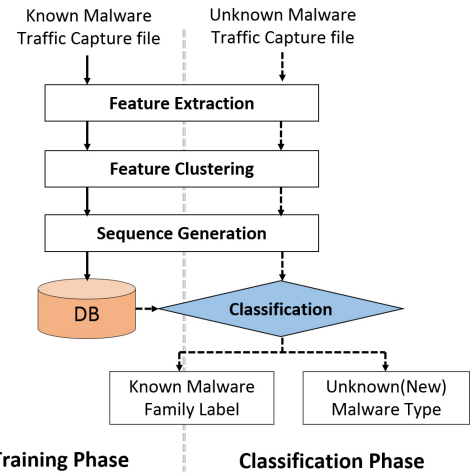

Fig. 1. Overview of malware classification

traffic flow generated by a dynamic analyzer during the execution of the malware. To meticulously classify malware on the basis of its behavior, the malware analysis system is recommended to suitably reflect malware activities. Furthermore, for effective classification the features must be easy to extract and provide sufficient information to discriminate between different malware families. We have obtained both the header and payload of network packets. However, when using the payload a lot of storage space and analysis time are required.

On the other hand, the header information of the packets can be analyzed even if the communication is encrypted. Accordingly, we decided to use the flow sequence that is most suitable for observing the flow of packets.

We use a dynamic analysis method to capture the network behavior of malware in pcap files. First, the malware samples are executed by dynamic analysis systems, and the pcap file is obtained from the result of the dynamic analysis. The pcap file is analyzed using Argus, which extracts flow data from network traffic files. Table 1 provides examples of flow data extracted from real malware samples. Note that IP addresses are sanitized for privacy protection. As shown in Table 1, the flow extracted by Argus contains all types of protocols of traffic that are invoked by the malware. Among these protocols, TCP and UDP can be deeply associated with the

Table 1. Examples of Flow Data

Dur	Seq	Proto	SrcAddr	DstAddr	Sport	Dport	State	Dir
2.999957	1	RARP	00:0c:29:89:7d:fa	00:0c:29:89:7d:fa			INT	who
0.000000	2	IGMP	10.0.0.0	224.0.0.1			INT	→
0.000332	3	ARP	192.168.1.1	192.168.1.2			CON	who
0.000000	1	RARP	00:0c:29:89:7d:fa	00:0c:29:89:7d:fa			INT	who
0.000000	1	RARP	00:0c:29:89:7d:fa	00:0c:29:89:7d:fa			INT	who
0.000000	1	RARP	00:0c:29:89:7d:fa	00:0c:29:89:7d:fa			INT	who
0.000000	1	RARP	00:0c:29:89:7d:fa	00:0c:29:89:7d:fa			INT	who
0.756285	4	UDP	192.168.1.2	10.0.0.1	1037	53	CON	↔
0.003504	5	TCP	192.168.1.2	*.*.*.212	1038	80	CON	→
0.000708	5	TCP	192.168.1.2	*.*.*.212	1038	80	RST	→
0.000131	6	UDP	192.168.1.2	192.168.1.255	138	138	INT	→
0.000012	6	UDP	192.168.1.2	192.168.1.255	138	138	REQ	→

behavior of malware, thus we adopt them as features for clustering.

A feature used for clustering is not a characteristic that only determines whether the packets are normal or malicious. Rather, it is used as a representative attribute of the flow element. Therefore, our method only requires appropriate extraction of the flow characteristics as a preprocessor. We defined 14 features as listed in Table 2.

First, the 5-tuple (SrcAddr, DstAddr, Sport, Dport, Proto) and direction (Dir) information, which can be automatically extracted from a packet header, are defined. Dir is a feature indicating the direction of information flow. If a flow does not include direction, it is defined as who. However, if a flow has direction, it is defined as →, ↔, or ←. The other features are the duration of the flow (Dur), the Argus sequence number from the particular session (Seq), the transaction state of the flow (State), the total, source, and destination transaction packet counts (TotPkts, SrcPkts, DstPkts), and the source and destination bits per second (SrcLoad, DstLoad).

Table 2. Features of Flow Data

Feature Name	Explanation
Dur	Record total duration of flow
Seq	Argus sequence number
Proto	Transaction protocol (e.g., TCP, UDP, ARP)
SrcAddr	Source IP address
DstAddr	Destination IP address
Sport	Source port number
Dport	Destination port number
Dir	Direction of transaction (e.g., →, ↔, ← who)
State	Transaction state (e.g., INT, CON, RST, REQ)
TotPkts	Total transaction packet count
SrcPkts	Src → Dst packet count
DstPkts	Dst → Src packet count
SrcLoad	Source bits per second
DstLoad	Destination bits per second

3.2 Feature Clustering with K-means Algorithm

To cluster flow data, we use a K-means algorithm that is commonly used for unsupervised learning techniques. In the work of Erman et al.[25], K-means is suitable for classifying traffic flows faster than other algorithms. It proceeds by selecting k initial cluster centers and then iteratively refining them. k is a positive integer number specifying the number of clusters and has to be given in advance. The four steps of the K-means clustering algorithm are:

- Select an initial partition with k cluster centers; repeat steps 2 and 3 until clusters stabilize.
- Initialize the k cluster centroids. This can be done by arbitrarily dividing all objects into k clusters, computing their centroids, and verifying that all centroids are different from one another. Alternatively, the centroids can be initialized to k arbitrarily chosen, different objects.
- Iterate over all objects and compute the distances to the centroids of all clusters. Assign each object

to the cluster with the nearest centroid.

• Recalculate the centroids of the modified clusters.

A distance function is required in order to compute the distance (i.e., similarity) between two objects. The most commonly used distance function is the Euclidean distance, defined as follows:

$$d(x,y)=\sqrt{\sum_{i=1}^{m}(x_i-y_j)^2} \quad (1)$$

where $X=\{x_1,x_2,\cdots x_{m-1},x_m\}$ and $Y=\{y_1,y_2\cdots y_{m-1},y_m\}$ are two input vectors with m quantitative features. In the Euclidean distance function all features contribute equally to the function value. However, since different features are usually measured with different metrics or at different scales, they must be normalized before applying the distance function.

The formula for the square error V is shown by Equation (2):

$$V=\sum_{i=1}^{k}\sum_{j\in S_j}|x_j-\mu_i|^2 \quad (2)$$

The square error is calculated as the square of the distance between each object x and the center of its cluster μ_i.

The clustering process applies the K-means clustering algorithm to datasets that contain the flow extracted as described in Subsection 1. The rationale of this approach is the assumption that the network behavior of malware forms different clusters in the feature space. The flow data may contain outliers that do not belong to a bigger cluster, yet this does not disturb the K-means clustering process as long as the number of outliers is small.

An essential problem of the K-means clustering method is to define an appropriate number of clusters k. As an initial value, we choose k to be in the range 6 to 20, assuming that the flow data of network behavior form many different clusters.

Table 3 shows an example of clustering for the condition $k=8$. In this example, the K-means algorithm is applied to a Botnet Watcher dataset, which consists of 91,441 flows generated by 766 malware samples. Clusters are assigned a label from A to H, as shown in Table 3.

3.3 Network Flow Sequence Pattern Generation

In this step, a sequence pattern for each variant of malware is generated using the clustering result. The sequence represents the extracted behavior of the malware.

When two sequence patterns are similar, the same network activity may be the cause. We can identify malware families that have distinctive network behavior. Table 4. shows an example of sequence patterns generated by K-means clustering with $k=8$. As shown in Table 4, similar sequences can possibly belong to the same malware family.

Table 3. Result of Flow Clustering ($k=8$)

Cluster	A	B	C	D	E	F	G	H	Total
Number of Flows	3,810	15,545	3,265	3,827	2,541	4,665	12,615	8,342	54,610
Percentage (%)	6.9	28.5	6.0	7.0	4.6	8.5	23.1	15.2	100

Table 4. Example of Sequence Data ($k=8$)

Malware name	Sequence data
Virus.Sality.gen.1	BGGBGGBBBGGBGGBGGBGGEEEEEEEEEBEEEEEEEEEEEEBE...
Virus.Sality.gen.2	CBGHBGGBBBGGBGGBGGBGGEEEEEEEEEEEEEEEEEEEECEBE...
Backdoor.Simda.abxr	DCCCDBAACAAADABAGGADDDDGGBDDCCBD
Backdoor.Simda.acak	DGHGGGHGGGGGGGGGHGGGGGDGGGGGGGGGGGGGGBGGHH...
Backdoor.Simda.acam	DGHGGGHGHHGGGGGGHGGGGGGGGGGGGGGGGGGGGBGB...
Backdoor.Simda.acbg	GHGGGHHHGGGGGGGHGGGGGCGGGGGGGGGGGGGGBGB...

Through the matching of the activity and sequence of the malware, we determine the efficiency of the method.

Figure 2 presents the sequence patterns generated by K-means clustering. The objective of *Trojan.Win32.AntiFW* is to modify system settings that allow or augment potential malware behavior. It downloads files and makes an attempt at registry modification. In the next step, therefore, the sequence patterns are used to identify the malware families.

Trojan.Win32.AntiFW	=	***BEEJGJGAAABEKKBEKKKABBBBEAAA***

Fig. 2. Sequence patterns generated by K-means clustering

3.4 Classification using Sequence Alignment

In this step, we classify malware samples using sequence alignment. As an example, we have aligned the malware samples *Virus.Sality.gen.1* and *Trojan-Spy.Win32.Zbot.rfjs* from Table 2 using two algorithms. The Needleman-Wunsch algorithm attempts to align every element in every sequence.

On the other hand, the Smith-Waterman algorithm attempts to align only part of the two sequences. Therefore, the Smith-Waterman algorithm obtains a shorter total alignment than the Needleman-Wunsch algorithm.

In both methods, the similarity between two sequences is given as follows:

$$Similarity = \frac{Length\ of\ subsequence\ with\ highest\ score}{Total\ length\ of\ sequence\ used\ by\ alignment}$$

These two algorithms have drawbacks when applied to our method. When the difference between the lengths of sequences is large, they are difficult to define as similar even if their similarity is 100%. In the so-called Unified algorithm, we calculate the average of the similarities obtained from the two algorithms. The similarity from the Unified algorithm is given as follows (where N refers to Needleman-Wunsch, S refers to Smith-Waterman, and U refers to the Unified algorithm):

$$Similarity = \frac{Similarity_N + Similarity_S}{2}$$

As an example, the results for the similarity between the variants of Sality and Zbot(Sality 1: *Virus.Sality.gen.1*, Sality 2: *Virus.Sality.gen.2*, Zbot: *Trojan-Spy.Win32.Zbot.rfjs*) are shown in Table 5. The results show that the Unified algorithm overcomes the shortcomings of the two individual algorithms.

Table 5. Example of Similarity Results (%)

	$Similarity_S$	$Similarity_N$	$Similarity_U$
Sality 1 & 2	91.2	91.0	91.1
Sality 1 & Zbot	94.7	5.6	50.2

Ⅳ. Experiments and Results

In this chapter we demonstrate the results of our experiments. We have experimented with various algorithms and data to classify malware using sequences of network behavior. In Section 1, we introduce the dataset used in the experiments of Sections 2. Section 2 presents an experiment that used malware collected from an actual Internet environment for one whole year. Finally, in Section 3 we discuss the result of experiment.

4.1 Data of Malware Samples and Family Labeling

We suppose that the malware samples were executed by a dynamic malware analyzer to collect network traffic capture files. Our method, therefore, adopts traffic data collected by Botnetwatcher[7], which has been developed by NTT Secure Platform Laboratories and connected to the Internet. This dataset consists of the network traffic (pcap file) gathered during a 30-minute execution of each malware sample using a dynamic malware analysis system. Many dynamic analysis systems have difficulty analyzing the activity of malware accurately because they operate under the controlled environment of the Internet. However, the Botnetwatcher used in our method conducted a

real-time analysis that can be associated with an open environment.

The dataset also includes labels assigned by eleven kinds of anti-virus software that scanned each malware sample. Of these, we used the labels from Kaspersky to create labeled datasets because the classification criteria applied by Kaspersky are based on the behavior of malware. Kaspersky has classified malware using all features of malware, including network traffic flow. In contrast, our method only focuses on the network behavior of malware extracted from network flow. In the experiment, we compared the classification of Kaspersky and our classification method. If the results of the comparison are similar to the labeling of Kaspersky, it is possible to prove the effectiveness of our classification based on network behavior.

To acquire the family name of malware programs, we use the naming rule of Kaspersky Anti-virus[1]. All objects detected by Kaspersky are named as follows:

[Prefix:]Behaviour.Platform.Name[.Variant]

The prefix identifies the sub-system that detected the object, but it is not an obligatory part of the name and may not be present. The behaviour specifies what the detected object does, and the platform is the environment in which the program code is executed. This can refer to both software and hardware. The name is the official name given to the detected object, which defines the family of detected objects. Finally, a variant is a modification of a detected object.

In the experiment, we regard [Prefix:]Behaviour.Platform.Name as the malware family name. For example, the family name of *Backdoor.Win32.Agent.a* is *Backdoor.Win32.Agent,* and *Backdoor.Win32.Agent* and *Trojan.Win32.Agent* are different family names. Because anti-virus venders use their own name rules, a situation in which different anti-virus venders assign the same family name to a malware program is rare.

4.2 Classification Experiment

In this section, we expand our dataset to include malware samples collected from October 2013 to October 2014 using Botnetwatcher mentioned in Section 1.

4.2.1 Training and Testing Datasets

We identified 45 families of malware samples using Kaspersky[2]. We divided them into training and testing datasets. We extracted approximately 10% of samples from each of the 45 families of malware samples in the training dataset (102 samples), and obtained the remaining samples from the testing dataset (664 samples). To obtain a balanced training dataset, we limited the distribution of each family in the training dataset to 40% of the entire dataset. Table 6 shows the number of families and samples in the training and testing datasets. The total number of samples is 766.

The number of malware that have another malware sample in the same family group less than 9 is 110. The number of malware that have another malware sample in the same family group more than 10 and less than 100 is 453. The number of malware that have another malware sample in the same family group more than 100 is 203.

Table 7 shows part of the similarity between the variants of *Backdoor.Win32.Tofsee*, which is one of the families we identified. In Table 7, the maximum and minimum values of the similarity are 91.27% and 76.63%, respectively, and their table cells are highlighted. The average similarity percentage between the variants of *Backdoor.Win32.Tofsee* is 84.62%.

Table 8 shows part of the similarity between the variants of *Backdoor.Win32.Simda*, which is one of the families we identified. In Table 8, the maximum

Table 6. Numbers for the Training and Testing Datasets

	Training	Testing	Total
# of Families	45	45	
# of Samples	102	664	766

1) http://www.securelist.com/en/threats/detect?chapter=136

2) http://www.kaspersky.com/

Table 7. Similarity between Variants of Backdoor.Win32.Tofsee (%, $k=16$)}

	Tofsee 1	Tofsee 2	Tofsee 3	Tofsee 4	Tofsee 5
Backdoor.Win32.Tofsee 1	-	-	-	-	-
Backdoor.Win32.Tofsee 2	86.14	-	-	-	-
Backdoor.Win32.Tofsee 3	87.51	91.27	-	-	-
Backdoor.Win32.Tofsee 4	81.57	76.63	78.27	-	-
Backdoor.Win32.Tofsee 5	84.21	90.32	87.46	78.36	-

Table 8. Similarity between Variants of Backdoor.Win32.Simda (%, $k=16$)}

	Simda 1	Simda 2	Simda 3	Simda 4	Simda 5
Backdoor.Win32.Simda 1	-	-	-	-	-
Backdoor.Win32.Simda 2	42.70	-	-	-	-
Backdoor.Win32.Simda 3	38.36	94.23	-	-	-
Backdoor.Win32.Simda 4	39.75	92.00	90.38	-	-
Backdoor.Win32.Simda 5	42.24	89.12	86.54	86.27	-

and minimum values of the similarity are 94.23% and 38.36%, respectively, and their table cells are highlighted. The average similarity percentage between the variants of *Backdoor.Win32.Simda* is 59.22%.

4.2.2 Classification Accuracy

We measured the similarity between the training and testing datasets, that is, we calculated the similarity between one testing dataset and each individual training dataset. Then, we sorted them by ranking from No.1 to No.102 in decreasing order of similarity and made comparisons using the labels assigned by Kaspersky. By taking the comparison results into account, the testing data are classified into the family with the highest degree of similarity. We define the following indices for performance comparison:

$$Classification\ \ Accuracy = \frac{\#\ of\ \ T\ classified\ correctly}{\#\ of\ \ T}$$

※ *where T represents malware samples of the testing dataset.*

The results are shown in Table 9. For the calculations, we used the three algorithms referred to in Chapter 3. We obtained similarities of 35.4% and 40.1% by using the Smith-Waterman and Needleman-Wunsch algorithms, respectively. By using the Unified algorithm, which is the average value of the two algorithms, we obtained the highest result of 40.4 %.

Table 9. classification Accuracy (%, $k=16$, $Length=100$)}

Algorithm	Smith-Waterman	Needleman-Wunsch	Unified
Accuracy	35.4	40.1	40.4

4.3 Discussion

Figure 3 indicates the classification accuracy with the three algorithms. According to Figure 3, when we take care of classification within the Top 3, the classification accuracy exceeds 65%. This result shows the feasibility whthin the Top 3, the classification accuracy exceeds 65%. This result shows the feasibility of our method by improving our algorithm.

On the other hand, *Trojan.Yakes* accounts for almost half the samples. Also, some families, such as *Trojan.Win32.Yakes* and *Trojan-Ransom.Win32.Foreign*, have much larger numbers of samples than the other families. This is because the dataset that we used is real malware collected from October 2013 to October 2014, focusing on a specific distribution of malware samples. For future work,

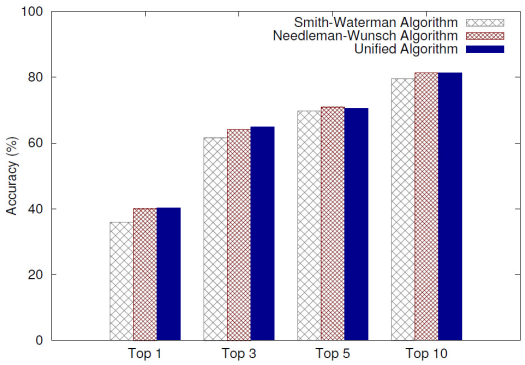

Fig. 3. Classification Accuracy by Ranking (%, $k=16$, $Length=100$)

we will complement the distribution of malware samples.

As shown in Table 9, the highest classification accuracy was obtained by the Unified algorithm, which is the average value of the Smith-Waterman and Needleman-Wunsch algorithms, and not by a method using only one algorithm. This shows that using the Unified algorithm can overcome the individual shortcomings of the two algorithms. We obtained a low accuracy, less than 40%, which we must devise measures to improve.

Finally, the method needs to be improved to classify unknown malware. In this study, we performed experiments with only known families. For future work, we need to improve the classification method so it can be used with new malware.

V. Conclusion

In this paper, we proposed a malware classification method based on sequence patterns generated from the network flow of malware samples. The goal was to classify malware by using only its network behavior.

The method begins by obtaining flow data from traffic extracted by a dynamic analysis of malware. We extract features of the flow and cluster them using a K-means algorithm. On the basis of the clustering result, sequence patterns are generated. These patterns represent the network behavior of a malware family. Finally, we classify the malware behavior by using a sequence alignment algorithm. In particular, we have explored malware behavior by adapting the Needleman-Wunsch and Smith-Waterman algorithms.

We evaluated our method with 766 malware samples and obtained a classification accuracy of approximately 40%. The average of the similarity between samples from the same malware family is 40.4% in the experiment using an expanded dataset. It is useful to detect malware behavior without having to deeply analyze its binary.

Our future work will focus on studying the classification of unknown malware against known malware families using network behavior. We intend to continue developing and testing the classification system while expending our malware samples and refining our classification algorithm. We will also collect and analyze malicious code identified by the author and reflect it in the our system. It will be possible to predict and identify attack groups. Furthermore, we will erect malicious code data sets for comparing and verifying their classification accuracy so that they can be compared with other classification methods.

References

[1] McAfee, *Mcafee labs threats report*, Nov. 2014.

[2] K. Rieck, et al., "Learning and classification of malware behavior," *DIMVA '08*, pp. 108-125, Paris, France, Jul. 2008.

[3] M. Bailey, et al., "Automated classification and analysis of internet malware," *Recent advances in Intrusion Detection*, vol. 4637, pp. 178-197, 2007.

[4] S. Cesare and Y. Xiang, "Malware variant detection using similarity search over sets of control flow graphs," *IEEE TrustCom*, pp. 181-189, 2011.

[5] J. Kinable and O. Kostakis, "Malware classification based on call graph clustering," *J. Comput. Virol.*, vol. 7, no. 4, pp. 233-245, 2011.

[6] M. K. Shankarapani, et al., "Malware detection using assembly and API call sequences," *J.*

Comput. Virol., vol. 7, no. 2, pp. 107-119, 2011.

[7] K. Iwamoto and K. Wasaki, "Malware classification based on extracted api sequences using static analysis," in *Proc. AINTEC '12,* pp. 31-38, Bangkok, Thailand, Nov. 2012.

[8] K.-H. Kim and M.-J. Choi, "Linear SVM-Based android malware detection and feature selection for performance improvement," *J. KICS*, vol. 39, no. 8, pp. 738-745, Aug. 2014.

[9] H.-H. Kim and M.-J. Choi, "Android malware detection using auto-regressive moving-average model," *J. KICS*, vol. 40, no. 8, pp. 1551-1559, Aug. 2015.

[10] U. Bayer, et al., "Scalable, Behavior-Based malware clustering," *NDSS Symp.*, vol. 9, 2009.

[11] I. K. Cho, et al., "Malware similarity analysis using API sequence alignments," *JISIS*, vol. 4, no. 4, pp. 103-114, 2014.

[12] G. Berger-Sabbatel and A. Duda, "Classification of malware network activity," *Multimedia Commun., Services and Security*, vol. 287, pp. 24-35, 2012.

[13] N. Stakhanova, M. Couture, and Ali A. Ghorbani, "Exploring network-based malware classification," *IEEE MALWARE*, Oct. 2011.

[14] R. Perdisci, W. Lee, and N. Feamster, "Behavioral clustering of HTTP-Based malware and signature generation using malicious network traces," *NSDI Proc. 7th USENIX Conf. Netw. Syst. Design and Implementation,* p. 26, San Jose, California, Apr. 2010.

[15] M. Z. Rafique, et al., "Evolutionary algorithms for classification of malware families through different network behaviors," *GECCO '14,* pp. 1167-1174, Vancouver, Canada, Jul. 2014.

[16] I. Ahmed and K. Lhee, "Classification of packet contents for malware detection," *J. Comput. Virol.*, vol. 7, no. 4, pp. 279-295, 2011.

[17] S. Nari and Ali A. Ghorbani, "Automated malware classification based on network behavior," *IEEE ICNC*, pp. 642-647, 2013.

[18] Y. Jung and M. Park, "Network defense mechanism based on isolated networks," *J. KICS*, vol. 41, no. 9, pp. 1103-1107, Sept. 2016.

[19] S. Coull, et al., "Intrusion detection: A bioinformatics approach," in *Proc. IEEE Annu. Comput. Security Appl. Conf.*, 2004.

[20] Scott E. Coull and Boleslaw K. Szymanski, "Sequence alignment for masquerade detection," *J. Computational Statistics & Data Anal.*, vol. 52, no. 8, pp. 4116-4131, Apr. 2008.

[21] M. K. Shankarapani, et al., "Malware detection using assembly and API call sequences," *J. Comput. Virol.,* vol. 7, no. 2, pp. 107-119, 2011.

[22] J. Pedersen, et al., "Fingerprinting malware using bioinformatics tools building a classifier for the zeus virus," in *Proc. Int. Conf. Security and Management (SAM)*, Jan. 2013.

[23] Saul B. Needleman and Christian D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *J. molecular biology*, vol. 48, no.3, pp. 443-453, Mar. 1970.

[24] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *J. Molecular Biology*, vol. 147, no. 1, pp. 195-197, Mar. 1981.

[25] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in Proc. MineNet '06 ACM, pp. 281-286, Pisa, Italy, Sept. 2006.

임 효 영 (Hyo-young Lim)

2009년 2월 : 육군사관학교 정보과학과 졸업

2015년 2월 : 일본 나고야국립대학 정보과학과 석사

2015년 3월~현재 : 국방부

<관심분야> 네트워크 플로우, 데이터마이닝, 머신러닝

김 완 주 (Wan-ju Kim)

1998년 2월 : 서울과학기술대학교 전자공학과 졸업
2008년 1월 : 국방대학교 전산정보학과 석사
2012년 3월~현재 : 아주대학교 NCW학과 박사과정
<관심분야> 사이버전, 정보보호, 전술통신

노 홍 준 (Hong-jun Noh)

2008년 2월 : 아주대학교 정보 및 컴퓨터공학과 졸업
2015년 2월 : 아주대학교 컴퓨터공학과 박사
2015년 3월~현재 : LIG넥스원 C4I연구소 통신연구센터 연구원
<관심분야> 위성 통신망, MF-TDMA, 랜덤 액세스, 전술 데이터링크

임 재 성 (Jae-sung Lim)

1983년 2월 : 아주대학교 전자공학 졸업
1985년 2월 : KAIST 영상통신 석사
1994년 2월 : KAIST 디지털통신 박사
1998년 3월~현재 : 아주대학교 컴퓨터공학과 정교수
<관심분야> 이동 및 위성통신, 무선네트워크, 국방전술통신