Cluster Comput (2019) 22:S921–S929
https://doi.org/10.1007/s10586-017-1110-2

Improvement of malware detection and classification using API call sequence alignment and visualization

Hyunjoo Kim[1] · Jonghyun Kim[1] · Youngsoo Kim[1] · Ikkyun Kim[1] · Kuinam J. Kim[2] · Hyuncheol Kim[3]

Received: 23 April 2017 / Revised: 24 July 2017 / Accepted: 10 August 2017 / Published online: 12 September 2017
© Springer Science+Business Media, LLC 2017

Abstract Conventional malware detection technologies have the limitation to detect malware because recent malware uses a variety of the avoidance techniques such as obfuscation, packing, anti-virtualization, anti-emulation, encapsulation technology in order to evade the detection of malware. To overcome this limitation, it is necessary to obtain new detection technology which is able to quickly analyze massive malware and its variants, and take the rapid response to cyber intrusion. Therefore in this paper, we proposed the malware detection and classification method and implementation of our system based on the dynamic analysis using the behavioral sequence of malware (API call sequence) and sequence alignment algorithm (MSA). Also we evaluated the effectiveness of our proposed method through the experiment.

✉ Kuinam J. Kim
kuinamj@gmail.com

✉ Hyuncheol Kim
hckim@nsu.ac.kr

Hyunjoo Kim
hjookim@etri.re.kr

Jonghyun Kim
jhkim@etri.re.kr

Youngsoo Kim
blitzkrieg@etri.re.kr

Ikkyun Kim
ikkim21@etri.re.kr

[1] Information Security Research Division, Electronics & Telecommunications Research Institute, Daejeon, Korea

[2] Department of Convergence Security, Kyonggi University, Suwon, Korea

[3] Department of Computer Science, Namseoul University, Cheonan, Korea

Keywords Malware detection and classification · Behavioral sequence · Similarity · Multiple sequence alignment · Visualization

1 Introduction

Conventional malware detection techniques are used to detect and analyze the characteristics of the malicious code. However, thousands, or tens of thousands of malicious codes are generated in a day. Thus, the gap between the number of emerging malware from attackers and the number of signatures that security vendors can deal with can be hardly narrowed, and the gap is growing rather steadily. Thus, we cant build a solid defense system from threat of malware in the way of existing detection technology. The new system or method which can supplement conventional detection methods is being introduced.

As mentioned above, malware is a major means of the recent cyber-attacks since there is a limitation to identify malware due to the avoidance technique to evade the malware detection.

Usually, malware is spreading through a variety of propagation methods and infecting a vulnerable system for the malicious purpose such as spam mail distribution, privacy disclosure, system corruption, and denial of service. Since malware can hinder and delay the malware analysis using a variety of techniques such as obfuscation, packing, anti-virtualization, anti-emulation, encapsulation technology, an attacker makes malware simply exit when malware is executed in a virtual machine or emulator. Also through the anti-debugger technology, malware generates unexpected events by detecting the presence of a debugger. Furthermore, the malware uses a method to modify the kernel-level structure and to conceal its own process as a typical technique of

impeding the detection technology. If you do not diagnose the infection of malware, the infected systems can continue to send your personal information to the malware distributors.

Recently, to evade the detection of malware based on the behavior of malware which especially uses API calls as the behavioral information, an attacker attempts to intentionally remove or distort the behavioral patterns of a malicious code by adding new API calls, and deleting or replacing existing ones that do not affect the functionality of the original code [1]. In addition to these limitations of the conventional detection techniques, the technology that analyzes a large amount of malware at the same time and identifies the new malware, as well as its variants, has not yet been widely developed.

Therefore, it is necessary to obtain new detection technology which is able to quickly analyze massive malware and its variants, and take the rapid response to the cyber intrusion. In this paper, we propose the malware detection and classification method based on the dynamic analysis and the implementation of the malware detection and classification system, which uses the API call sequences and multiple sequence alignment (MSA) algorithm [7]. Since the API call sequences of malware can indicate the overall behaviors of malware, our proposed method conducts extraction and analysis of the API call sequences by monitoring and hooking the system API calls in the real PC not in a virtual machine. Our proposed method also applied an MSA algorithm in order to produce the representative behavioral patterns (namely, behavioral sequence chains) for some clustered malware families.

The rest of this paper is organized as follows. We review the related work on malware detection using the sequence alignment based on the dynamic analysis in Sect. 2. Then we propose the methods of generation of malware behavioral sequence chain and our malware detection and classification based on API call sequence alignment in Sect. 3. In Sect. 4, we depict the visualization scenes for the analysis result obtained from our proposed system. In Sect. 5, we evaluate the detection performance of our proposed method through the experiment. Finally, we conclude in Sect. 6.

2 Related work

We introduce some studies on malware detection using the sequence alignment or longest common subsequence (LCS) algorithm [9] related to our proposed method.

Cho, et al. [2] proposed a malware similarity calculation system to detect variants of malware among the same group of malware and suggested the process which can diminish overheads of the sequence alignment processes. They removed the repeated API subsequences from the whole API call sequence to reduce the total length of it in order to improve the overall accuracy of the similarity calculation. Such a system can detect malware by computing similarity of the API call sequence of a target process with all malware sequences stored previously because it does not have the common patterns of the malware groups/families. Therefore, they evaluated their proposed system by similarity calculation experiments and verified the performance improvement of the alignment process by eliminating the repeated API subsequences.

Also Chen, et al. [3] proposed a new approach to detect malware by using multiple sequence alignment techniques to align variable length of virus and worm code. It shows that converting the hexadecimal code of viruses and worms to the amino acid alphabet and the rational numbers between 0 to 1 could be effective to detect malware. They also demonstrated that their novel approach would be feasible to identify malware code through multiple sequence alignment followed by data mining methods such as ANNs and symbolic rule extraction. However, this study was aimed to identify the malware signature with the hexadecimal code instead of the API call sequences.

Kim, et al. [1] proposed a method for detecting code clones, software plagiarism, code theft, and polymorphic malware using sequence alignment algorithms. They show that an attacker can evade the detection by inserting dummy codes or replacing existing codes in the sequence. They found that these kinds of polymorphic attacks were ineffective and the birthmark sequences extracted from the malicious program with a hybrid approach based on "non-consecutive insertion" and "highest frequency deletion" were proved to be effective. This study also discussed the limitation of sequence alignment techniques for comparing software birthmarks and demonstrated the performance of sequence alignment algorithms against polymorphic attacks.

Elhadi, et al. [4] proposed a malware detection system to enhance malware detection using API call graph. Their proposed mechanism consists of the following phases; in the preprocessing phase, they unpacked malware and extracted the API calls using API call monitoring tool so that the API call and its parameters were stored in the database. In the second phase, they constructed the data dependent API call graph and updated the malware API call graph database for matching algorithm. In the last phase, the system compared the data dependent API call graph by graph matching algorithm and calculated the similarity in order to identify whether the input sample is malware or not. Usually, graph matching algorithms have a NP-Complete problem and are very slow due to their computational complexity. Therefore, their proposed system simplified the data dependent API call graph to reduce the computation complexity of graph matching by selecting paths with the same edge label in the query graph and the data graph. Then, their system calcu-

lated the similarity using LCS algorithm by mapping two paths selected from both the query graph and the data graph. They showed that their system achieved the performance on 98.6% detection rates and 98.8% accuracy for their dataset (75 malware samples, 10 benign programs).

3 The proposed system based on behavioral sequence chain

To detect and classify known/unknown malware and its variants, in this paper, a malware detection and classification system based on the behavioral sequence chains of malware is proposed. As shown in Fig. (1), Our proposed system is composed of five functional steps: (1) Data collection and Sequence extraction; (2) Feature extraction and preprocessing; (3) Clustering; (4) Behavioral sequence chain extraction; and (5) eDetection and classification.

In the first step, we executed all malware samples in the limited environment and extracted API call sequences by hooking API calls. Our 1790 malware samples of our dataset were collected from the famous website providing malware such as malshare.com [10] and VXVolt.net [11]. In the second step, we selected the features which can identify a target process as benign or malware and converted the integer sequence of API calls extracted into the character sequence. Because MSA algorithm can use only character sequence. In the third step, in order to generate precisely the behavioral sequence chains of the malware families, we categorized the collected malware samples as some malware groups in advance. Then we were able to generate the behavioral sequence chain of malware by extracting the common API call sequence of the final malware groups (namely, families) clustered in the previous step. Finally, we detected and classified malware by calculating similarity value between the behavioral sequence chains and API call sequence of a target process.

Figure 2 shows two procedures of the proposed system - the behavioral sequence chain generation and malware detection and classification, which include the functional steps in Fig. 1 and are explained in each Sects.3.1 and 3.2.

3.1 Malware behavioral sequence chain generation

In this section, we explain the procedure to generate the behavioral sequence chains identifying the malware families. As shown in Procedure 1 of Fig. 2, this procedure consists of four functional steps explained in Fig. 1. Through the sequence extraction and feature extraction and preprocessing, the behavioral sequences of malware samples are extracted by API hooking library after they are executed in the limited execution environment. In the feature extraction and preprocessing and clustering, these sequences are made in the form of a character sequence used in the MSA algorithm and are grouped into some families with similar behavior. From the behavioral sequence chain extraction step, each behavioral sequence chain of the malware family is generated by extracting the common sequence with the MSA algorithm. Finally, each behavioral sequence chain is added in the database.

- *API call sequence extraction* We executed all malware samples in the limited environment which are not the virtual environment but real PC. Whenever we executed them, in order to recover the infected PC to its clean state, we used PC Recovery software, called Wow-Comback. We extracted API call sequences from malware samples with API hooking library. These sequences are composed of 47 classes (from 1 to 47) which 102 API calls hooked are classified into according to their functions. At this point, we found that the interval time for collecting API calls is about 0.5 s. When we checked the functional running time of malware, most of the malware generally terminated their malicious function within 0.5 s except for particular malware such as the logic bomb that is triggered by responding to an event or when a certain date/time is reached. For most of the malware in our dataset, the length of the API call sequences represented the number of features collected for 0.5 s. It also implied meaningfully the length of the API call sequences collected during execution time of a process as shown in Fig. 3.
- *Feature extraction and preprocessing* We defined 26 classes used as a feature in advance. Theses features must be able to identify a target process as benign or malware. In this step, we extracted feature sequence composed of 26 classes from the API call sequence generated in the previous step. And we converted the integer sequence into the character sequence to use it as an input sequence of MSA as shown in Fig. 2.
- *Clustering* We grouped the collected malware samples into some malware families with similar behavior. This clustering is necessary in order to generate the behavioral sequence chains of the malware families. Because the noise caused by dissimilarity in MSA makes a unprecise alignment result and the significant performance degradation of data processing of MSA. Therefore we had three-steps clustering mechanism. The first step is the division by decision tree (Weka C4.5) [12]. Through this step, we can get 48 malware groups clustered by 856 malware samples. The second step is the subdivision by the MSA algorithm and we can get several subgroups according to the sequence patterns of each group generated in the first step. Finally, we merged the similar subgroups by calculating similarity between the common behavioral sequence of each subgroup.

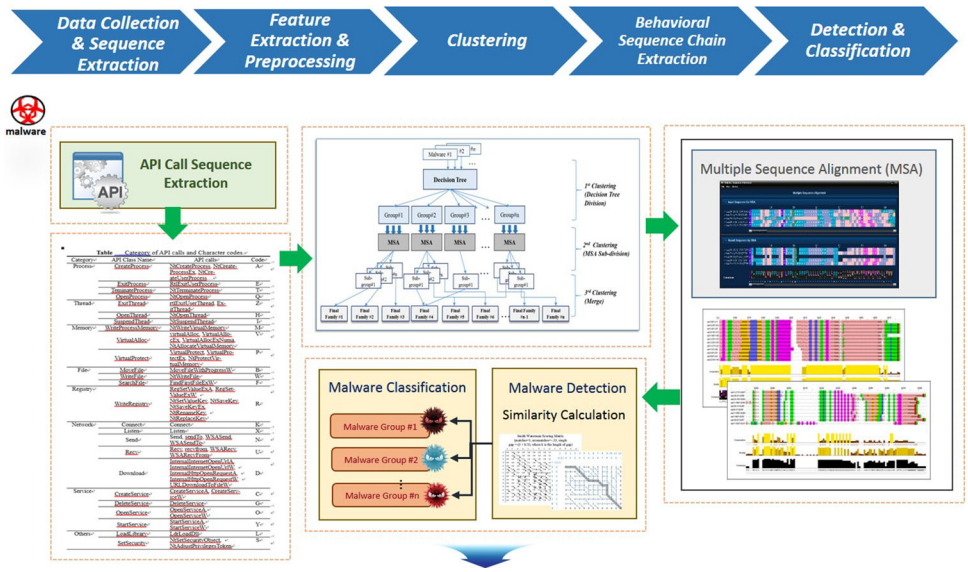

Fig. 1 Functional flow of the proposed system

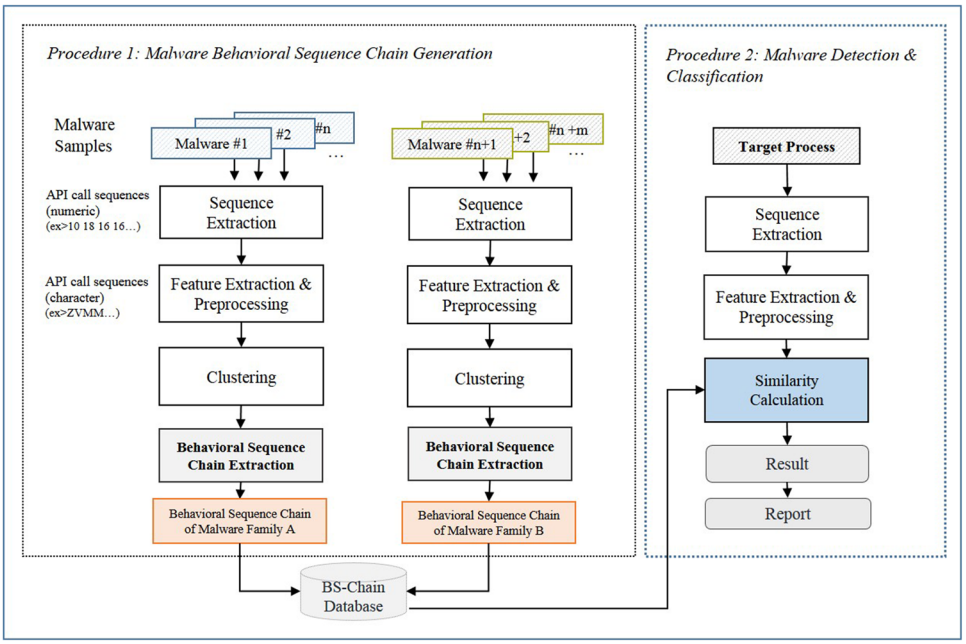

Fig. 2 Two procedures of the proposed system

- *Behavioral sequence chain extraction* We generated the behavioral sequence chain by extracting the common behavioral sequence of each final malware family generated in the clustering step. For this, we used ClustalX [5,9] tool and JalView application widely used for MSA algorithm in bioinformatics.

3.2 Malware detection and classification

In this section, we discuss the malware detection and classification scheme by calculating the similarity value between the behavioral sequence of a target process and the behavioral sequence chain of each malware family. In order to calculate similarity, we firstly generated the behavioral sequence of a target process through the sequence extraction and feature extraction and preprocessing as shown in Procedure 2 of Fig. 2. Then we calculated the similarity value between two sequences and compared them. Once the sequence of a target process has the similarity value over than the similarity threshold, we detect the target process as malware and classify it into the malware family with the highest similarity value. The sequence of a target process detected and classified as malware is reported and stored in the database.

Our proposed system used two algorithms to calculate the similarity value, which is Smith-Waterman and LCS algorithm. Because they are based on sequence alignment and provide the matched region and the length of the longest common sequence.

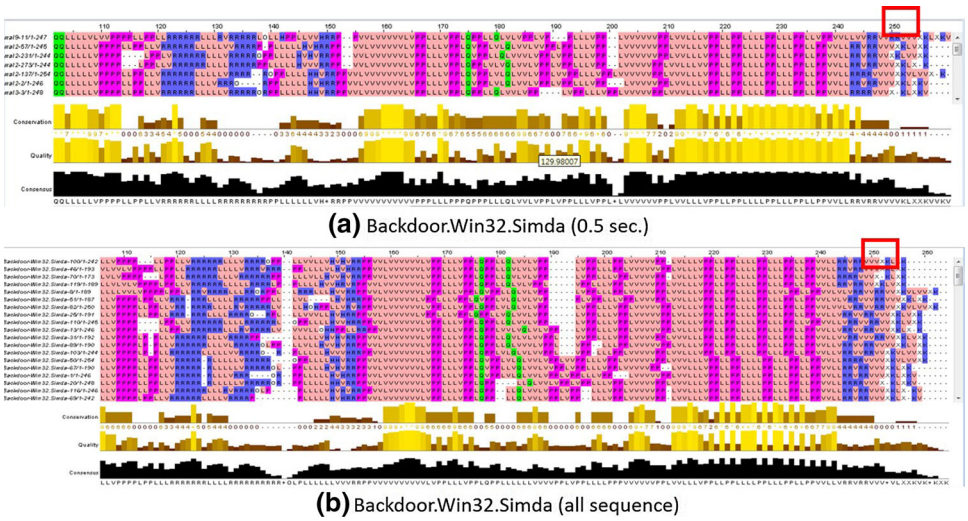

Fig. 3 Comparison of the Sequence Length of Backdoor.Win32.Simda

3.2.1 Similarity based on Smith-Waterman algorithm (similarity-SW)

The Smith Waterman algorithm is a popular algorithm of some local alignment methods [6]. It defines that alignment algorithm is to identify regions of similarity between two strings of sequences by inserting or deleting the defined gap into each sequence. The similarity between two sequences using Smith-Waterman is calculated as follows;

$$Sim_{sw}(X,Y) = \begin{cases} 0, & if\ X_{Length} = 0\ or\ Y_{Length} = 0 \\ N/A, & if\ X_{Length} > Y_{Length} \\ \frac{Match_{sw}(X,Y)}{X_{Length}} \times 100, & if\ X_{Length} \leq Y_{Length} \end{cases}$$
$$X_{Length} = length(X),\ Y_{Length} = length(Y) \quad (1)$$

Let sequence X and sequence Y denote the behavioral sequence chain of each malware family compared and the behavioral sequence of a target process, respectively.

Where, $Match_{sw}(X,Y)$ is matching count of all matching regions of the alignment result from the Smith-Waterman algorithm without the gap count. As shown in Eq. (1), the length of the sequence X must not be longer than of the sequence Y. In other words, the sequence of a target process must be compared with all behavioral sequence chains which have the smaller length than itself. It is in order to decide whether the sequence of a target process contains the behavioral sequence chain having high similarity value.

3.2.2 Similarity using LCS (similarity-LCS)

We define the similarity using LCS as follows: Let sequence X and sequence Y represent each the behavioral sequence chain of the malware families compared and the sequence of a target process.

$$Sim_{LCS}(X,Y) = \begin{cases} 0, & if\ X_{Length} = 0\ or\ Y_{Length} = 0 \\ N/A, & if\ X_{Length} > Y_{Length} \\ \frac{LCS(X,Y)}{X_{Length}} \times 100, & if\ X_{Length} \leq Y_{Length} \end{cases} \quad (2)$$

where $LCS(X,Y)$ is the length of the longest common subsequence between two sequences. The condition statement of the sequence length of this equation is equal to that of the Eq. (1). We will discuss the performance measurement and comparison with some methods including these two methods in the next Sect. 5.

4 Visualization of the proposed system

This section describes several visualization scenes of the analysis results which the proposed system provides. Fig. 4 shows the implementation of visualizing both the behavior of the process executed in a computer system and the behavioral features obtained from computer systems (namely, hosts) and network devices. The behavioral features of a host process can be categorized by eight types of function and the behavioral features of network process can be categorized by four protocol types of the packets occurred in its network.

The right of Fig. 4 visualizes the behavioral sequence of each process according to the order of occurrence with the sequence of behavioral features that are already represented on 3D DNA structure shown in left side of Fig. 4. Through this visualization technique, a security manager can recognize the behavioral features and sequences of the processes running on the host. In addition, once receiving the alert generated by a security analyzer that detects the abnormal behavior from an external host or a network device, the information related

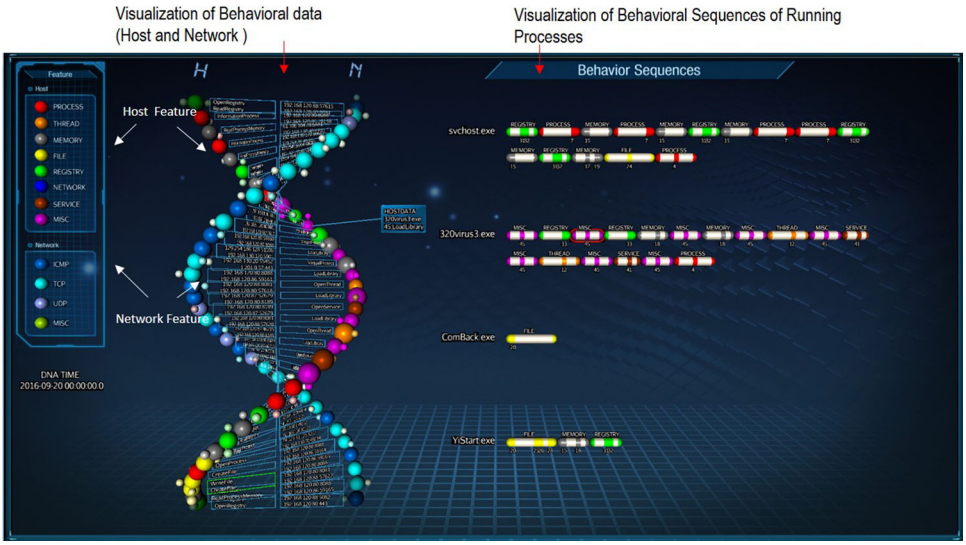

Fig. 4 Scene of behavior visualization

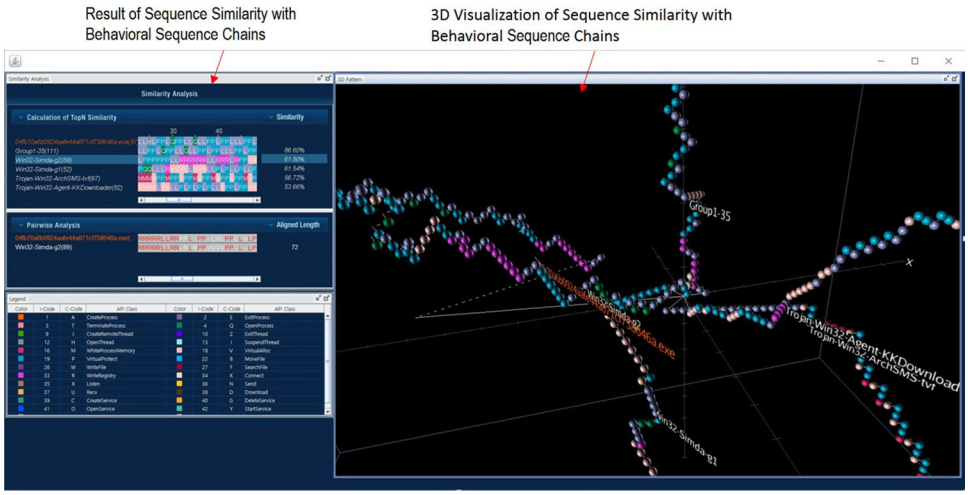

Fig. 5 Visualization scene of similarity analysis

to the alert can be displayed on the scene of both 3D DNA structure and behavioral sequence in Fig. 4.

Through our visualization technique, we may intuitively represent the behavioral sequence of the process running on the host and also analyze the correlation of behavioral features obtained from the particular host and the associated network data.

Figure 5 shows the result of analyzing similarity between the behavioral sequence of a target process (unknown suspicious process) and the behavioral sequence chains of the defined malware families. Furthermore, it displays the analytical results belonging to the top 5 of the highest similarity value. That is, the left side of Fig. 5 shows the behavioral sequence of a target process and each behavioral sequence chain of five malware families belonging to the top 5 of the highest similarity value, and displays the results of the pairwise alignment which is used for calculating similarity using LCS algorithm. 3D visualization scene in right side of Fig. 5 represents the behavioral sequence of a target process and the behavioral sequence chains in the three-dimensional coordinate space according to the results of similarity analysis on the left side of Fig. 5. It is more intuitive and easier to provide a security manager with the results of similarity since it displays a similar region between the behavioral sequence of a target process and each behavioral sequence chain of five malware families belonging to the top 5 of the highest similarity value.

We are developing MSA application based on a ClustalX and JalView to extend the number of features in the future work. As shown in Fig. 6, we can extract the most common subsequence. Each character of the common subsequence is extracted when the occupancy ratio of the character in each column of multiple sequences from the MSA result is over a certain threshold. We can assign this threshold optionally.

Fig. 6 MSA application developed for behavioral sequence chain generation

5 Evaluation and empirical results

In this section, we explain the method of determining a similarity threshold and evaluate performance detecting malware of the proposed system. As a dataset, we used 1790 malware and 1138 benign API call sequences. 854 of malware sequences were used for generating the behavioral sequence chains of malware families and the remainder was for measuring detection performance. To evaluate the effectiveness of the proposed system, we use Recall, Precision, Accuracy and F-measure.

$$\text{Recall (R)} = TP/(TP + FN) \tag{3}$$

$$\text{Precision (P)} = TP/(TP + FP) \tag{4}$$

$$\text{Accuracy (AC)} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

$$\text{F} - \text{Measure (F)} = 2 \times \frac{(Recall \times Precision)}{(Recall + Precision)} = \frac{2TP}{2TP + FP + FN} \tag{6}$$

- TP (True Positive) is the number of running processes correctly detected as malware.
- TN (True Negative) is the number of running processes correctly detected as benign.
- FP (False Positive) is the number of running processes mistakenly detected as malware.
- FN (False Negative) is a number of running processes mistakenly detected as benign.

5.1 Evaluation and empirical results

A suitable similarity threshold should be selected to identify malware and benign executables more accurately. For determination of this threshold, we have several experiments by increasing a similarity threshold in increments of 5 from 70% to 95%.

As shown in Table 1, F-measure is the highest at 94.30 when the threshold is 85%. Through these experimental results, we can determine that a suitable similarity threshold for our test data is 85%.

Table 1 Comparison of detection performance by changing similarity threshold

Similarity threshold (%)	Recall (%)	Precision (%)	Accuracy (%)	FP (%)	F-measure
95	57.26	99.29	81.82	0.35	72.63
90	88.89	98.22	94.26	1.32	93.32
85	93.59	95.01	94.89	4.04	94.3
80	97.12	83.62	90.12	15.64	89.87
75	99.25	72.07	82.3	31.63	83.5
70	99.35	53.2	60.27	71.88	69.29

Table 2 Comparison of detection perfomance

Method	Actual class	Detected class		R (%)	P (%)	AC (%)	FP (%)	F
		Malware	Benign					
Similarity- SW	Malware	866 (TP)	59 (FN)	93.62	92.91	93.81	6.02	93.26
	Benign	66 (FP)	1030 (TN)					
Similarity-LCS	Malware	876 (TP)	60 (FN)	93.59	95.01	94.89	4.04	94.3
	Benign	46 (FP)	1092 (TN)					

5.2 Comparison of detection performance

In this section, we evaluated detection performance of our proposed methods (the Similarity-SW and the Similarity-LCS) explained in Sect. 3.2.

As shown in Table 2, two methods have similar performance at recall. However, similarity-LCS has higher performance at a precision (95.01%), accuracy (94.89%), and the false positive rate (4.04%). Therefore, we calculated the f-measure of them and we can see that the similarity-LCS method achieved higher performance than the similarity-SW method.

6 Conclusion

In this paper, we proposed the malware detection and classification system by generating the behavioral sequence chains of some malware families and calculating the similarity between the behavioral sequence chain and the sequence of a target process. This system consists of two procedures and five functional steps. First, to generate the behavioral sequence chain, we executed the collected malware samples in the limited execution environment and extracted their API call sequences. Through the feature extraction and preprocessing and clustering steps, these sequences were processed as an input sequence set of MSA. From the result of MSA, the behavioral sequence chain of malware is finally generated and added in the database. Second, to detect a target process as malware or benign, we extracted the API call sequence of a target process and converted it into a character sequence. Then we detected and classified malware by measuring the similarity between the API call sequence of a target process and the behavioral sequence chains of the malware families. At that time, this similarity value was calculated from our proposed equation based on an LCS algorithm.

Also, we introduced visualization for the analysis result obtained from our proposed system. Through this visualization, a security manager will be able to recognize the threat situation intuitively and easily. Finally, we evaluated detection performance of our proposed methods and the method of calculating similarity based on an LCS algorithm achieved higher performance on the precision, accuracy, false positive rate and F-measure (95.01, 94.89, 4.04, and 94.30%).

Acknowledgements This work was supported by Institute for Information and communications Technology Promotion (IITP) Grant funded by the Korea Government (MSIP) (No. 2016-0-00078, Cloud-based Security Intelligence Technology Development for the Customized Security Service Provisioning).

References

1. Kim, H., Khoo, W., Li, P.: Polymorphic attacks against sequence-based software birthmarks. In Proceeding of 2nd ACM SIGPLAN Workshop on Software Security and Protection (2012)
2. Cho, I., Kim, T., Shim, Y.J., Park, H., Choi, B., Im, E.: Malware similarity analysis using API sequence alignments. J. Internet Serv. Inf. Secur. **4**(4), 103–114 (2014)
3. Chen, Y., Narayanan, A., Pang, S., Tao, B.: Multiple sequence alignment and artificial neural networks for malicious software detection. Proceedings of 8th International Conference on Natural Computation (ICNC), pp. 261–265. May 2012
4. Elhadi, A., Maarof, M., Barry, B.: Improving the detection of malware behavior using simplified data dependent API call graph. Int. J. Secur. Appl. **7**(5), 29–42 (2013)
5. Thompson, J.D., Gibson, T.J., Higgins, D.G.: Multiple sequence alignment using ClustalW and ClustalX. Curr. Protoc. Bioinform. Chapter 2: Unit 2.3 (2002)
6. Polyanovsky, V., Roytberg, M., Tumanyan, V.: Comparative analysis of the quality of a global algorithm and a local algorithm for alignment of two sequences. Algorithms Mol. Biol. **6**(1), 25 (2011)
7. Multiple Sequence Alignment.: Internet: http://www.ebi.ac.uk/Tools/msa/
8. Longest common subsequence problem, Wikipedia, Internet: https://en.wikipedia.org/wiki/Longest_common_subsequence_problem
9. Clustal: Multiple Sequence Alignment, Internet: http://www.clustal.org/
10. The MalShare Project.: http://malshare.com
11. VXVolt.: http://vxvault.net
12. WEKA Open Sources tools for Data Mining.: http://www.cs.waikato.ac.nz/ml/weka/

Hyunjoo Kim received her M.S. degree and Ph.D. degree in Computer Engineering from Sungkyunkwan University, Korea, in 2002 and 2016, respectively. She joined the Electronics and Telecommunications Research Institute (ETRI) in 2002. She is currently working as a senior member of the engineering staff of the Intelligent Security Research Group. Her research interests include the malware analysis, network security, cloud computing and Big-Data analytics.

Jonghyun Kim received the M.S. degree and the Ph.D. degree in computer science from the University of Oklahoma, USA, in 2000 and 2005, respectively. He was a researcher with the Samsung Electronics in 1995-1997 and the Samsung SDS in 2000. He joined the Electronics and Telecommunications Research Institute (ETRI) in 2005. He is also involved in standardization activities as an associate rapporteur of ITU-T SG17/Q.4(cybersecurity). His research interests include Information Security, Cyber Security, Network Forensics, and Network Security Function Virtualization.

Youngsoo Kim received his Ph.D. degree in the department of computer engineering from Sungkyunkwan University, Korea, in 2009. From 2000, he has been a principal engineer in Electronics and Telecommunications Research Institute (ETRI), Daejon, Korea. From 2012 to 2015, he was an adjunct professor in the department of computer science & engineering in Chungnam National University, Korea. His research interests include information security, cryptography, network security, cloud computing, and digital forensics.

Ikkyun Kim received his M.S. and Ph.D. degrees in Computer Engineering from the Kyoung-Pook National in 1996 and 2009 respectively. He joined the Electronics and Telecommunications Research Institute in 1996. He was an invited researcher for the Purdue University from 2004 to 2005. He is currently working as a director of the Network Security Research Laboratory of the ETRI. His research interests are on the Network Security, Computer Network, Cloud Security and BigData Analytics.

Kuinam J. Kim is a professor of Information Security Department in the Kyonggi University, Korea. He received his Ph.D. and MS in Industrial Engineering from Colorado State University in 1994. His B.S in Mathematics from the University of Kansas. He is Executive General Chair of the Institute of Creative and Advanced Technology, Science, and Engineering. His research interests include cloud computing, wireless and mobile computing, digital forensics, video surveillance, and information security. He is a senior member of IEEE.

Hyuncheol Kim received his Ph.D. degree in the department of information engineering from Sungkyunkwan University, Korea, in 2005. From 1992 to 2002, he was senior engineer in Electronics and Telecommunications Research Institute (ETRI), Daejon, Korea. He has been a professor in the department of computer science in Namseoul University, Korea. His research interests include cloud computing, next generation network and embedded computing systems.