IEICE TRANS. INF. & SYST., VOL.E94–D, NO.11 NOVEMBER 2011

PAPER Special Section on Information and Communication System Security

A Novel Malware Clustering Method Using Frequency of Function Call Traces in Parallel Threads

Junji NAKAZATO[†a)], **Jungsuk SONG**[†], **Masashi ETO**[†], **Daisuke INOUE**[†], ***and*** **Koji NAKAO**[†], ***Members***

SUMMARY With the rapid development and proliferation of the Internet, cyber attacks are increasingly and continually emerging and evolving nowadays. Malware – a generic term for computer viruses, worms, trojan horses, spywares, adwares, and bots – is a particularly lethal security threat. To cope with this security threat appropriately, we need to identify the malwares' tendency/characteristic and analyze the malwares' behaviors including their classification. In the previous works of classification technologies, the malwares have been classified by using data from dynamic analysis or code analysis. However, the works have not been succeeded to obtain efficient classification with high accuracy. In this paper, we propose a new classification method to cluster malware more effectively and more accurately. We firstly perform dynamic analysis to automatically obtain the execution traces of malwares. Then, we classify malwares into some clusters using their characteristics of the behavior that are derived from Windows API calls in parallel threads. We evaluated our classification method using 2,312 malware samples with different hash values. The samples classified into 1,221 groups by the result of three types of antivirus softwares were classified into 93 clusters. 90% of the samples used in the experiment were classified into 20 clusters at most. Moreover, it ensured that 39 malware samples had characteristics different from other samples, suggesting that these may be new types of malware. The kinds of Windows API calls confirmed the samples classified into the same cluster had the same characteristics. We made clear that antivirus softwares named different name to malwares that have same behavior.

key words: *malware analysis, behavior of malware, clustering*

1. Introduction

With the rapid development and proliferation of the Internet, increasing numbers of cyber attacks threaten critical computer networks and systems and continue to emerge and evolve. Malware – a generic term for computer viruses, worms, trojan horses, spywares, adwares, and bots – is one of the most typical and lethal security threats. Since malwares are usually operated through cooperation with others in an automated manner, they frequently lead to serious security incidents that can cause significant damages not only to end users, but also to the Internet's infrastructure itself.

To cope with the security threats caused by malwares, research has previously focused on two main areas: macroscopic and microscopic. The macroscopic approach monitors networks in real time and focuses on more effectively understanding the latest trends in malicious activity over a wide range of networks. Numerous projects are underway around the world and several monitoring systems make their monitoring reports publicly available [1]–[11]. The microscopic approach, on the other hand, focuses on analyzing executable malware codes captured by honeypots, etc., in order to obtain a deep understanding of their characteristics and behaviors [12]–[18].

Though these approaches have been studied and deployed in various analysis systems, there has been no effective correlation analysis between the two. However, considering the close relationships between global phenomena observed from the macroscopic approach and their root causes analyzed in the microscopic approach, a hybrid approach for such analysis is essential. To this end we have been developing the Network Incident analysis Center for Tactical Emergency Response (nicter) [19]–[21], which incorporates both approaches. The nicter integrates results obtained from both macroscopic and microscopic analysis to obtain useful and practical insight on malware activity.

During the correlation analysis, since identifying the characteristics and behaviors of each malware is time-consuming, one of the most important challenges is minimizing the time needed to analyze malwares so that results can be obtained in real time. To further complicate matters, a great deal of unknown malwares, or variants of known malwares, emerge dynamically each day. In order to mitigate the workload for malware analysis, previous research has focused on clustering of malwares varieties according to their similarities. Anubis [13], for instance, applies a clustering method to group malwares wherein members in the same group are similar from the standpoint of their overall behavior. If the number of function calls, their order, and their types that were executed by malwares are similar, they are assigned as members of the same group.

In many cases, however, malware simultaneously performs multiple processes and threads during its execution period; thus the order, number, and type of function calls it executes depends heavily on the execution environment and timing. Therefore, there is a possibility of misclassifying such types of malware into different groups in Anubis even if they contain the same function calls. Situations in which attackers try to confuse malware analyzers by crafting malwares with embedded unrelated (or noisy) functions also make Anubis less effective.

This paper presents a new classification method to overcome the limitations of Anubis, to enable more effective clustering of malwares. Our classification method consists of two main techniques – N-gram and TF-IDF. The frequency of an API sequence consisting of Windows API

Manuscript received February 1, 2011.
Manuscript revised June 3, 2011.

[†]The authors are with National Institute of Information and Communications Technology, Koganei-shi, 184–8795 Japan.
a) E-mail: nakazato@nict.go.jp

DOI: 10.1587/transinf.E94.D.2150

calls is calculated by using the N-gram technique. The characteristics of malware samples are deduced by using the TF-IDF technique. We evaluated our classification method using 2,312 malware samples, and were able to classify them into 93 groups. More than 90% of all samples were classified into at most 20 clusters, and 39 samples (about 1.7%) were dissimilar to the others. We compared the results with the results of name-based clustering by three types of antivirus softwares, and confirmed that our classification method could correctly classify the samples by their characteristics.

The rest of the paper is organized as follows. Section 2 provides a brief description of the Micro Analysis System (MicS) that is a part of nicter. Section 3 details our classification method, and Sect. 4 gives experimental results and analysis before the paper concludes.

2. Related Work

2.1 Malware Classification

In general, there are two types of malware analysis methods; one is static analysis and another is dynamic analysis. While the static analysis tries to analyze program code embedded in malware, the dynamic analysis focuses on analyzing behavior of malware by executing it on an analysis environment. Since most recent malwares are packed for the analysis avoidance and analysts need to manually unpack malwares to obtain their behaviors, it is not easy to analyze malware with high accuracy in the static analysis. On the other hand, the dynamic analysis can provide a capability of automatic analysis, and obtain internal and external behaviors of malwares such as Windows API sequences, network activities. Since there are a number of emerging malware samples nowadays, it is important to automatically classify them based on their behavior.

Bailey et al. proposed a new malware classification scheme that calculates distance between malware samples based on profiles of malware samples derived by "Normalized Compression Distance (NCD)" [30]. Then it used pairwise single-linkage clustering, which defines the distane between two clusters as the minimum distance between any two members of the clusters. Finally, they showed the tradeoff between the number of clusters and the average of cluster size by depth of tree-cutting.

Rieck et al. proposed malware classification scheme based on character string of Windows API call [31]. In order to classify a malware into "known" or "unknown", it learned the Windows API call of each family name by using SVM. Actually, they classified about 10,000 malware samples into 14 families. As a result of classification, the total average of accuracy was 88% and the accuracy of three families became almost 100%.

Although these methods are able to recognize unknown behaviors of malware samples, there are some problems in the practical use of them. For instance, in [31], it always requires pre-learning of a new malware family when it is emerged at the first time. This means that the calculation cost is high and it is inefficient when using it for the long period. In [30], it is also inefficient, because it needs calculation of the distance between all malware samples and an additional malware sample. Then, we propose the malware classification method with a low additional cost.

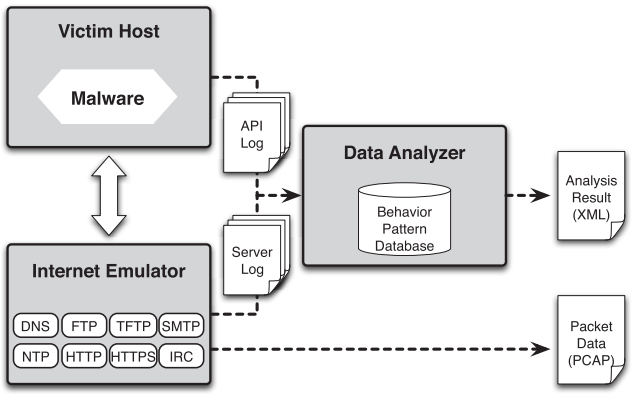


Fig. 1 Structure of micro analysis system.

2.2 MicS: Micro Analysis System

The MicS [22]–[24] is for conducting fully automated in-depth examination of malware in order to grasp its characteristics and activity, and to accumulate analysis results on a variety of malware. Figure 1 shows three main parts of the MicS: Victim Host, Internet Emulator, and Data Analyzer. The Victim Host executes a given malware and gathers an array of external information caused by its activity. It is capable of hooking and monitoring Windows API function calls used by malware. Meanwhile, files and registries accessed by the malwares are also observed. The Victim Host then outputs to the Data Analyzer logs including Windows APIs, files, and registries associated with the malware.

The Internet Emulator provides virtual Internet access for the Victim Host since most recent malwares variety attempt to communicate with external systems in order to propagate their infections. The Internet Emulator emulates various kinds of servers, such as DNS, FTP, TFTP, HTTP, SMTP, and IRC. All queries of the target malware in the Victim Host are sent to the Internet Emulator. The Internet Emulator then sends back apparent replies to the malware. It outputs all the logs of its own dummy servers to the Data Analyzer. It also records all packets transmitted between the Internet Emulator and the Victim Host as a pcap [25] file, which includes scan activity of the malware as well as all queries and replies with the dummy servers.

The Data Analyzer analyzes all logs sent from the Internet Emulator and the Victim Host. It can translate tedious low-level logs into high-level behavior patterns, taking advantage of the behavior pattern database, which stores many such malware patterns. New patterns can easily be added to the database. The analysis result is eventually created in XML format.

```
1: [2010/02/09 16:07:37.593] ovdkhjv.exe:1640, File, Call::ReadFile,
    [C:\WINDOWS\system32\ovdkhjv.exe][12d53c][800][12fe40][0], 0
2: [2010/02/09 16:07:37.593] louvez.exe:536, File, Call::ReadFile,
    [C:\WINDOWS\system32\ovdkhjv.exe][12d53c][2800][12fe40][0], 0
3: [2010/02/09 16:07:37.593] ovdkhjv.exe:1640, File, ReadFile,
    [C:\WINDOWS\system32\ovdkhjv.exe][12d53c][800][12fe40][0], 1
4: [2010/02/09 16:07:37.593] louvez.exe:536, File, ReadFile,
    [C:\WINDOWS\system32\ovdkhjv.exe][12d53c][2800][12fe40][0], 1
5: [2010/02/09 16:07:37.593] ovdkhjv.exe:1640, File, Call::WriteFile,
    [C:\DOCUME~1\chiyoda\LOCALS~1\Temp\yva1.tmp][12d53c][800][12fe40][0], 0
6: [2010/02/09 16:07:37.593] louvez.exe:536, File, Call::WriteFile,
    [C:\DOCUME~1\chiyoda\LOCALS~1\Temp\yva1.tmp][12d53c][2800][12fe40][0], 0
7: [2010/02/09 16:07:37.625] ovdkhjv.exe:1640, File, WriteFile,
    [C:\DOCUME~1\chiyoda\LOCALS~1\Temp\yva1.tmp][12d53c][800][12fe40][0], 1
8: [2010/02/09 16:07:37.703] louvez.exe:536, File, WriteFile,
    [C:\DOCUME~1\chiyoda\LOCALS~1\Temp\yva1.tmp][12d53c][2800][12fe40][0], 1
9: [2010/02/09 16:07:37.718] lwtczm.exe:384, Reg, Call::RegOpenKeyExA,
    [80000001][Software\Microsoft\Windows\CurrentVersion\Explorer][0][20019][77d80000], 0
10: [2010/02/09 16:07:37.718] louvez.exe:536, File, Call::ReadFile,
    [C:\WINDOWS\system32\louvez.exe][12d53c][2800][12fe40][0], 0
11: [2010/02/09 16:07:37.718] louvez.exe:536, File, ReadFile,
    [C:\WINDOWS\system32\louvez.exe][12d53c][2800][12fe40][0], 1
12: [2010/02/09 16:07:37.718] lwtczm.exe:384, Reg, RegOpenKeyExA,
    [80000001][Software\Microsoft\Windows\CurrentVersion\Explorer][0][20019][77d80000], 0
```

Fig. 2 Example of *API log* obtained from MicS.

In this paper, we utilize *API logs* obtained from the Victim Host to calculate the similarity between threads that malware executes. Figure 2 shows an example of *API logs* in which each row shows "[Analyzed date] Process name:Thread ID, API name, API function name, Argument values, return value." In the first row, for example, Analyzed date is "2010/02/09 16:07:37.593," Process name is "ovdkhjv.exe," Thread ID is "1640," API name is "File," API function name is "ReadFile" (the prefix "Call::" means called timing). Of particular note, our classification method uses "API name" and "API function name" to classify malware samples.

3. Proposed Classification Method

We classify malwares using their behavior characteristics. Malware's function is composed of Windows API calls such as `CreateFile`, `WriteFile`, `Write_Data` and so on. This means that malwares with different behavior contain a different API sequence. In our classification method, therefore, we use these API sequence as a criterion to classify a set of malwares into subsets.

The following subsections detail our classification method. We first calculate the frequency of each API sequence in malwares. Second, we give a TF-IDF score to each API sequence, which is used for representing the characteristic of malwares. Finally, we classify malwares according to the TF-IDF scores among them.

3.1 Overall Procedure

The MicS records a history of Windows APIs called by malwares into the *API log*. In general, malwares execute multiple *processes* and *threads* at the same time, it is important to extract API sequences within each thread in order to identify more exact behavior of malwares. We define three algorithms to classify malwares: (a) Calculating Frequency (*Calculation*), (b) Extracting Characteristic (*Extraction*), and (c) Classifying Malware (*Classification*).

(a) *Calculation***:** calculates the frequency of the API sequence, which includes *N* Windows API calls. These frequencies and API sequence are inputted into database (in Fig. 3 (a)).

(b) *Extraction***:** extracts the characteristics of a malware sample using the frequency of the API sequence. It obtains the frequency of the API sequence from database (in Fig. 3 (b)).

(c) *Classification***:** classifies malware samples into groups with the same API sequence characteristic. It makes use of the characteristics and the API sequence from the result of *Extraction* and database respectively (in Fig. 3 (c)).

In the first step, the API sequence that includes *N* Windows API calls is extracted from *API log* to calculate the frequency of the API sequence by the *Calculation* algorithm. In the second step, we extract the characteristics of malwares based on the frequency of the API sequence by the *Extraction* algorithm. We consider a method that malwares characteristics depends not only on frequency but also a specific API sequence. Finally, the *Classification* algorithm compares characteristics among each type of malware, and classifies those with the same characteristic.

The dynamic analysis (the MicS) consists of the Victim Host, the Internet Emulator, and the Data Analyzer. We implemented the experimental environment on MacPro for

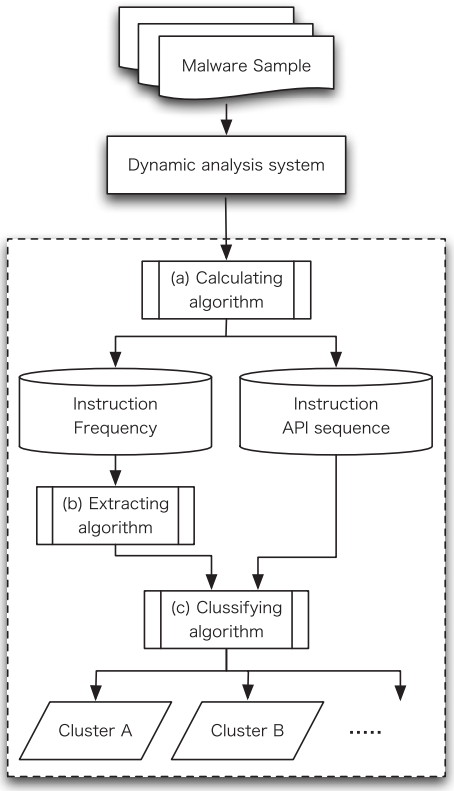


Fig. 3 Flowchart of proposed method.

Table 1 Simulation environments.

Victim Host	CPU	Pentium D 3.0 GHz
	OS	Windows XP (SP2)
Internet Emulator	CPU	–†
	OS	CentOS 4.3 (2.6.9-34)
Data Analyzer	CPU	Pentium D 3.0 GHz
	OS	Windows XP (SP2)
Experiment environment	CPU	Xeon 2.26 GHz (Dual)
	OS	MacOS 10.6.7

†The Internet Emulator is offered in the virtual environment of Victim Host using virtualbox.

the experiment of the proposed classification method. These simulation environments are shown in Table 1.

3.2 Calculating Frequency of API Sequence

A malware function can be shown by a sequence of Windows API calls. Therefore, we define a sequence of N Windows API calls as an API sequence, and calculate its frequency (tf_i) using the N-gram method. The N-gram calculates the frequency of N successive characters (or words). In general, two successive characters are called bi-gram, and three successive characters are called tri-gram. In our method, the API sequence is created from N successive Windows API calls. Therefore N-gram can be calculated from frequency of API sequence.

An *API log* is composed of *threads* in that each *thread* consists of numerous API sequences. Each thread ($T = \{t_1, \ldots, t_m\}$) includes API sequences ($S = \{s_1, \ldots, s_k\}$) such that $t_i \subseteq S$. The *Calculation* algorithm calculates the frequency of each API sequence by counting the number of occurrences of the API sequence s_i in *API log*. Note that if there are two identical threads (*e.g.* $t_i = t_j$ where $i \neq j$), one is not included in the frequency calculation. In other words, threads that have mutually different API sequences are used to calculate frequency.

Figure 4 (a) shows an example of calculating the frequency of an API sequence. There are two *threads* t_1 and t_2 with four and five Windows API calls, respectively. The *thread* t_1 consists of Windows API sequences such as "File, Call::ReadFile", "File, Call::WriteFile", "Reg, Call::RegOpenKeyExA" and "Net, Call::connect", and the *thread* t_2 consists of Windows API sequences such as "File, Call::ReadFile", "File, Call::WriteFile", "Reg, Call::RegOpenKeyExA", "Net, Call::connect" and "Net, Call::connect". These make three ($t_1 = \{s_1, s_2, s_3\}$) and four API sequences ($t_2 = \{s_1, s_2, s_3, s_4\}$), respectively ($N = 2$). Note that the API sequence s_1 consists of Windows API such as "File, Call::ReadFile" and "File, Call::WriteFile". The frequency of each API sequence calculates as s_1: $tf_1 = 2/7$, s_2: $tf_2 = 2/7$, s_3: $tf_3 = 2/7$, and s_4: $tf_4 = 1/7$.

3.3 Extracting Malware Characteristics

The *Extraction* algorithm extracts malware characteristics using the Term Frequency–Inverse Document Frequency (TF-IDF) method. TF-IDF is one method for comparing characteristics of many terms over many documents [32]. TF calculates the term's occurrence frequency. If the term appears in many documents, it is a general term even if the occurrence frequency was high. IDF is a measure of general importance that the term with high frequency appears only to a specific document. Therefore, the TF-IDF value of term that is appear a lot in a specific document is high, and this term shows characteristics of documents. We define the malware characteristic in the API sequence with a high TF-IDF score. Consequently we extract the API sequence using these compositions.

Figure 4 (b) shows an example that extracts the malware characteristics. There are two *threads* (t_1 and t_2) which contains three and four API sequences. Here, three API sequences (s_1, s_2, and s_3) are common in two *threads*, so there are four kinds of API sequences (s_1, s_2, s_3, and s_4) in this malware. Therefore three API sequences (s_1, s_2, and s_3) can be regarded as a general sequence. The inverse document frequency (IDF) is calculated as

$$idf_i = \log \frac{|T|}{|\{t_j : t_j \ni s_i\}|}$$

where $|T|$ is the number of all *threads*, and $|\{t_j : t_j \ni s_i\}|$ is the number of *threads* where the API sequence s_i appears. Then, the IDFs of three API sequences (s_1, s_2, and s_3) are $idf_{\{1,2,3\}} = \log \frac{2}{2} = 0$, and s_4 is $idf_4 = \log \frac{2}{1} \simeq 3/10$.

The TF-IDF score is calculated as the product of TF

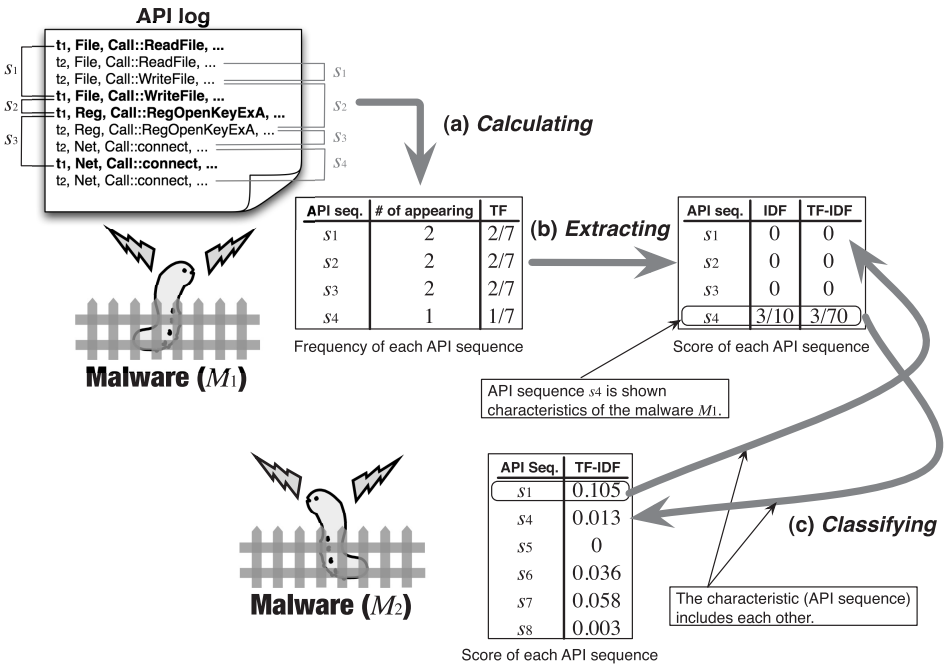


Fig. 4 Overall procedure.

and IDF. The TF-IDF scores of s_1, s_2, s_3, and s_4 are calculated to 0, 0, 0, and 3/70, respectively. In this case, the frequency of the API sequence s_4 is the lowest, but this malware was characterized by the API sequence s_4 whose TF-IDF score is high because it had appeared only for *thread* t_2. On the other hand, even though the frequency is high, a general API sequence (s_1, s_2, s_3) does not appear to a characteristic because they appear to both *threads*.

3.4 Classifying Malwares

The *Classifying* algorithm compares the characteristics of each malware sample. When both malwares contain each n API sequences that appears the characteristics, they are classified into the same group. If malwares classified into the same cluster, they include all API sequences that appear in the characteristics of each malware.

In Fig. 4 (c) shows an example that classifies two malware samples (M_1, M_2) into the same group. There are four API sequences (s_1, s_2, s_3, and s_4) and six API sequences (s_1, s_4, s_5, s_6, s_7, and s_8) in the malware samples M_1 and M_2, respectively. The same cluster is composed by comparing the n API sequences that appear to the characteristics of each malware sample. Therefore M_1 and M_2 are classified into the same group, because M_1 includes API sequence s_1 that has the characteristics of M_2 and M_2 includes the API sequence s_4 that has the characteristics of M_1 ($n = 1$).

4. Experimental Results

This section shows the experimental results of our classification method using 2,317 malware samples with unique hash values. We compare the relation between result of antivirus softwares and our scheme by using a *precision* and a *recall*. If both the *precision* and the *recall* are 1.0, it means that result of antivirus and our method are equivalent. Following subsection, we present classification result of antivirus softwares in Sect. 4.1, and our classification result is shown in Sect. 4.2. The accuracy of our scheme is shown in Sect. 4.3.

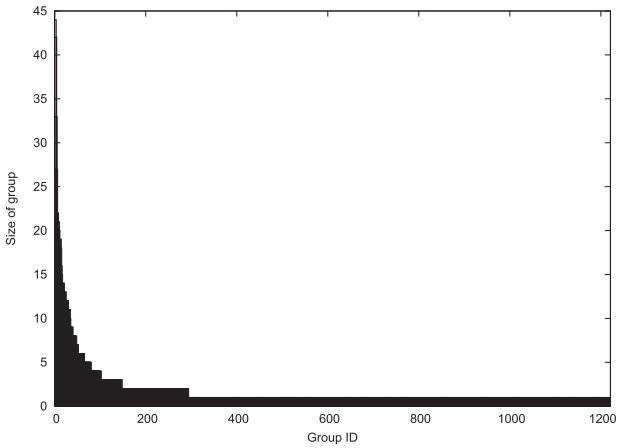


Fig. 5 Size of each group using result of name-based clustering.

4.1 Description of Experimental Data

We collected 2,317 malware samples whose hash values are unique in our honeypot. Symantec [27], McAfee [29], and Trend Micro [28] respectively determined them as 67, 166, and 331 kinds of groups. Also, malware samples were classified into 1,221 groups according to the combination of three antivirus softwares. We define these classification results as name-based clustering. Figure 5 shows the size

Table 2 The top 10 classification results using antivirus software: Norton AntiVirus (v. 14; Symantec Corp.), VirusScan (v. 13.15; McAfee Inc.), and Virus Buster (v. 2009; Trend Micro Inc.).

Malware #	Norton AntiVirus	VirusScan	Virus Buster
44	W32.Korgo.V	W32/Virut.gen	Mal_Korgo
42	W32.Korgo.V	W32/Virut.gen	PE_VIRUT.D-1
33	W32.Korgo.V	W32/Virut.gen.a	PE_VIRUT.AV
27	W32.Korgo.V	W32/Korgo.worm.v	Mal_Korgo
22	W32.Korgo.V	W32/Virut.gen.a	Mal_Korgo
22	W32.Korgo.V	Unknown	Malware.36DB2E5C
21	W32.Korgo.V	W32/Pate.b	PE_PARITE.A
21	W32.Korgo.V	W32/Virut.gen	PE_VIRUT.D-4
20	W32.Korgo.V	W32/Pate.a	PE_PARITE.A
19	W32.Spybot.Worm	W32/Sdbot.worm	Mal_Korgo

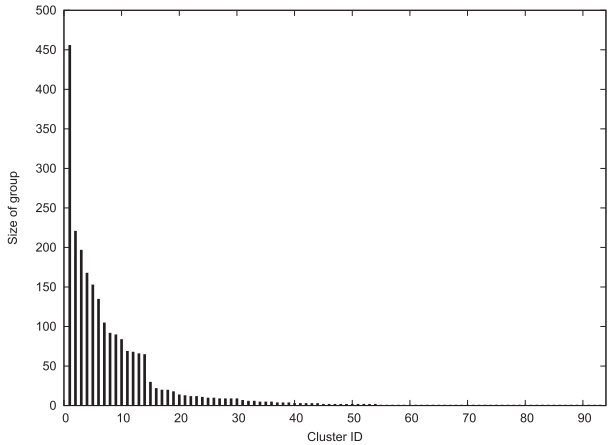


Fig. 6 Size of each cluster using the proposed classification method: graph shows the malware total included in each cluster.

(i.e., number of members) of each group, and the top 10 classification results are shown in Table 2. There are the variety of malware names in each antivirus software. Especially, the malwares were named different name such as W32/Virut.gen and W32/Pate.a and so on by VirusScan, even if Norton AntiVirus named the same name such as W32.Korgo.V. Therefore, there is contradiction in classification of malwares.

From Fig. 5, we can see that the largest group contained 44 malware samples, and about 40% (929) belonged to the independent group: they were dissimilar to the others.

4.2 Clustering Results of Proposed Scheme

We composed the API sequence of 10 Windows API calls ($N = 10$), and the malware characteristic is composed of five API sequences ($n = 5$). N and n determined the value from the experience. Malware samples were classified into 93 groups, with the largest cluster containing 456 samples.

Figure 6 shows the number of samples in each cluster. More than 90% of all malware samples were classified into at most 20 clusters, and 39 samples (about 1.7%) were dissimilar to the others.

4.3 Evaluation of Proposed Scheme

In order to verify superiority of the proposed method, it is important to compare it with other methods fairly. To this end, we need to prepare the existing dynamic analysis environments used in other methods and obtain the analysis results using the same malware samples. Since there are many different types of dynamic analysis environments [30], [31], it is extremely difficult to construct all of them, and consequently it is unable to carry out the direct comparison among the malware classification methods. Because of this, in previous research, the performance comparison was performed by the name-based clustering described in Sect. 4.1. Therefore, in this paper, we also evaluate the proposed method based on the name-based clustering.

For evaluation, we first inspected the *precision* and *recall* of our methods using the name-based clustering results. *Precision* shows the accuracy of our result by calculating an equation as follows:

$$precision = \frac{number\ of\ common\ malware\ samples\ (R)}{number\ of\ malware\ samples\ in\ our\ cluster\ (C)},$$

in which R is number of common malware samples in our clustering and in name-based clustering, and C is the number of malware samples in our clustering. *Recall* shows how much our clustering covers name-based clustering by a calculating equation as follows:

$$recall = \frac{number\ of\ common\ malware\ samples\ (R)}{number\ of\ malware\ samples\ in\ name\text{-}based\ (G)},$$

in which G is the number of malware samples in name-based clustering (show Fig. 7). If the *precision* is high and the *recall* is low, it means that the malware samples with the same name have different behavior. The other hands, if the *precision* is low and the *recall* is high, it means that the malware samples with the same behavior have different name.

Figure 8 shows the *precision* and *recall* of each cluster. *Precisions* were lower because the cluster contained many of name-based groups. However, most of clusters excluding 16 clusters (ID: 1, 2, 4, 5, etc) showed *recalls* of 100%. Therefore, most clusters contain complete groups of the name-based cluster. The *precision* and the *recall* are reversed on the right side of Fig. 8 (cluster ID is over 55). The

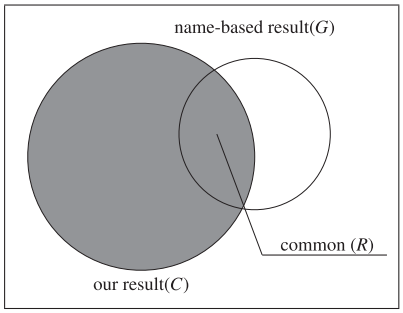


Fig. 7 Example of *precision* and *recall*: Note that $R = C \cap G$.

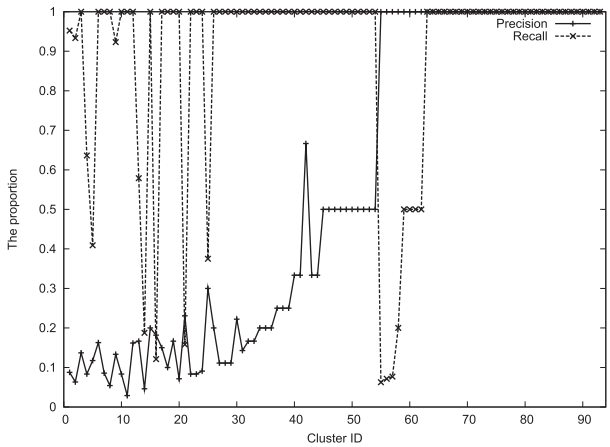


Fig. 8 *Precision* and *recall* of each cluster.

precision of these clusters became 100% because the cluster size was one, but the *recall* was lowered because this malware sample was part of a name-based cluster. The average of *precision* and the *recall* are about 55% and about 90%. It means that most of the name-based cluster are classified into the cluster of our scheme without dividing. However, the antivirus softwares give different name to the malwares which have the same behavior.

To confirm the accuracy of our clustering method, we investigated functions of malware samples in cluster ID 5. Here, we pulled up some malware samples randomly to compare their features. These were chosen as follows:

a) From $\bar{C} \cap G$
Three malware samples classified into the same group by name-based clustering and into different clusters with our clustering.
b) From R
Three malware samples classified into the same group by name-based clustering and into the same cluster with our clustering.
c) From $C \cap \bar{G}$
Four malware samples classified into a different group by name-based clustering and the same cluster with our clustering.

The characteristics of these malware samples were compared by the type of Windows API calls. Table 3 shows

Table 3 Comparative result of function of each malware sample.

Rule		HTTP access	Transaction of network error	Creation of backdoor	Modification of hosts file
a)	G_1	✓	✓	✓	✓
	G_2	–	–	–	✓
	G_3	✓	–	✓	✓
b)	R_1	–	–	–	–
	R_2	–	–	–	–
	R_3	–	–	–	–
c)	C_1	–	–	–	–
	C_2	–	–	–	–
	C_3	–	–	–	–
	C_4	–	–	–	–

✓: means the malware sample has it's function.
– : means the malware sample does not have it's function.

the characteristics of each malware sample's function. Note that there are many kinds of Windows API calls besides the functions to show in Table 3. The kinds of Windows API calls that are not shown in the Table 3 were the same. Three samples (G_1, G_2, G_3) had different characteristics. These samples were classified into different clusters by our method even if they were classified into the same group by name-based clustering. Our clustering technique proved capable of appropriately classifying by obtaining different characteristics. On the other hand, seven samples $(R_1, R_2, R_3, C_1, C_2, C_3, C_4)$ that were classified into the same cluster had the same characteristics, though three samples (R_1, R_2, R_3) were classified into the above name-based clustering group. Notably, four samples classified into different groups by name-based clustering could be classified into the same cluster. Therefore, our clustering scheme was able to classify by using the malware characteristics.

5. Conclusion

We proposed a new classification method that focuses on the behavior of individual threads invoked by the original process. We classified malware samples into clusters according to the characteristics of individual malware samples. We determined these characteristics from a sequence of Windows API calls. We achieved classification of 2,312 different malware types into 93 groups.

The classification let us resultantly obtain the trends of different types of the malware. In fact, 90% of the malware samples used in the experiment were classified into 20 groups at most. Moreover, it ensured that 39 malware samples had characteristics different from other samples, suggesting that these may be new types of malware. To confirm the accuracy of our clustering method, it compared with name-based clustering by using the *precision* and the *recall*. The average of *precision* and the *recall* were about 55% and 90%. In other words it means that antivirus softwares named different name to the same malwares.

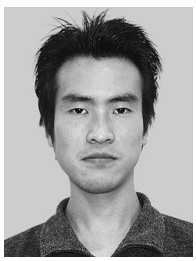
The accuracy of the clustering result of [30] is influenced by the parameter of the tree-cutting algorithm, and [31] is influenced by the kind of the family set used to learn. However, our classification method can decide the number

of clusters automatically without pre-learning process of a malware familly.

Our method doesn't need calculating for the comparison, because only the characteristics of malware sample is compared with. Therefor, it efficiently obtains useful information from an enormous amount of results. This offeres particular benefits in applying countermeasures against malware since it enables easy identification of different malware types.

References

[1] M. Bailey, E. Cooke, F. Jahanian, J. Nazario, and D. Watson, "The Internet motion sensor: A distributed blackhole monitoring system," 12th Annual Network and Distributed System Security Symposium (NDSS05), 2005.
[2] D. Moore, "Network telescopes: Tracking denial-of-service attacks and Internet worms around the globe," 17th Large Installation Systems Administration Conference (LISA '03), USENIX, 2003.
[3] V. Yegneswaran, P. Barford, and D. Plonka, "On the design and use of Internet sinks for network abuse monitoring," 7th International Symposium on Recent Advances in Intrusion Detection (RAID 2004), LNCS 3224, pp.146–165, 2004.
[4] SANS Internet Storm Center, http://isc.sans.org/
[5] REN-ISAC: Research and Education Networking Information Sharing and Analysis Center, http://www.ren-isac.net/
[6] Leurrecom.org Honeypot project, http://www.leurrecom.org/
[7] National Cyber Security Center, Korea, http://www.ncsc.go.kr/eng/
[8] Telecom Information Sharing and Analysis Center, Japan, https://www.telecom-isac.jp/
[9] IT Security Center, Information-Technology Promotion Agency, Japan, https://www.ipa.go.jp/security/index-e.html
[10] Japan Computer Emergency Response Team Coordination Center, http://jpcert.jp/isdas/index-en.html
[11] @police, http://www.cyberpolice.go.jp/english/obs_e.html
[12] C. Willems, T. Holz, and F. Freiling, "Toward automated dynamic malware analysis using CWSandbox," IEEE Security & Privacy Magazine, vol.5, no.2, pp.32–39, 2007. http://www.cwsandbox.org/
[13] Anubis: Analyzing Unknown Binaries, http://anubis.iseclab.org/
[14] U. Bayer, C. Kruegel, and E. Kirda, "TTAnalyze: A tool for analyzing malware," 15th Annual Conference of the European Institute for Computer Antivirus Research (EICAR), 2006.
[15] F. Bellard, "QEMU, a fast and portable dynamic translator," 2005 USENIX Annual Technical Conference, FREENIX Track, pp.41–46, 2005.
[16] NORMAN Sandbox Information Center, http://www.norman.com/microsites/nsic/
[17] C. Seifert, R. Steenson, I. Welch, P. Komisarczuk, and B. Endicott-Popovsky, "Capture - a behavioral analysis tool for applications and documents," 7th Annual Digital Forensic Research Workshop (DFRWS), 2007.
[18] Joebox, http://www.joebox.org/
[19] K. Nakao, K. Yoshioka, D. Inoue, M. Eto, and K. Rikitake, "Nicter: An incident analysis system using correlation between network monitoring and malware analysis," 1st Joint Workshop on Information Security (JWIS06), pp.363–377, 2006.
[20] K. Nakao, K. Yoshioka, D. Inoue, and M. Eto, "A novel concept of network incident analysis based on multi-layer observations of malware activities," 2nd Joint Workshop on Information Security (JWIS07), pp.267–279, 2007.
[21] K. Nakao, D. Inoue, M. Eto, and K. Yoshioka, "Practical correlation analysis between scan and malware profiles against zero-day attacks based on darknet monitoring," IEICE Trans. Inf. & Syst., vol.E92-D, no.5, pp.787–798, May 2009.
[22] D. Inoue, K. Yoshioka, M. Eto, Y. Hoshizawa, and K. Nakao, "Automated malware analysis system and its sandbox for revealing malware's internal and external activities," IEICE Trans. Inf. & Syst., vol.E92-D, no.5, pp.945–954, May 2009.
[23] Y. Hoshizawa, M. Morii, and K. Nakao, "A proposal of automated malware behavior analysis system," IEICE Technical Report, ICSS2006-07, 2006.
[24] D. Inoue, M. Eto, K. Yoshioka, Y. Hoshizawa, R. Isawa, M. Morii, and K. Nakao, "Micro analysis system for analyzing malware code and its behavior on nicter," The 2007 Symposium on Cryptography and Information Security (SCIS2007), 2F2-1, 2007.
[25] TCPDUMP public repository, http://www.tcpdump.org/
[26] D. Fetterly, M. Manasse, M. Najork, and J. Wiener, "A large-scale study of the evolution of web pages," Proc. 12th International Conference on World Wide Web (WWW '03), pp.669–678, 2003.
[27] Symantec Corp., http://www.symantec.com/
[28] Treand Micro Inc., http://www.trendmicro.com/
[29] McAfee Inc., http://www.mcafee.com/
[30] M. Bailey, J. Oberheide, J. Andersen, and Z.M. Mao, "Automated classification and analysis of Internet malware," 10th International Symposium on Recent Advances in Intrusion Detection (RAID 2007), LNCS 4637, pp.178–197, 2007.
[31] K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov, "Learning and classification of malware behavior," Detection of Intrusion and Malware, and Vulnerability Assessment (DIMVA 2008), LNCS 5137, pp.108–125, 2008.
[32] L.-P. Jing, H.-K. Huang, and H.-B. Shi, "Improved feature selection approach TFIDF in text mining," International Conference on Machine Learning and Cybernetics (ICMLC 2002), pp.944–946, 2001.



Junji Nakazato received his B.S. and M.S. degrees in Electrical Engineering from Tokai University, Japan in 2001 and 2003, respectively. He received his Ph.D. degree in the Graduate School of Engineering, Tokai University, Japan in 2006. He is currently a researcher at National Institute of Information and Communications Technology (NICT), Japan. His research interests include network security, malware analysis, cryptography theory, and privacy preserving technologies. He is a member of IPSJ.

Jungsuk Song received his B.S. and M.S. degrees in Information and Telecommunication Eng. from Korea Aerospace University, Korea in 2003 and 2005, respectively. He received his Ph.D. degree in the Graduate School of Informatics, Kyoto University, Japan in 2009. He is currently a researcher at National Institute of Information and Communications Technology (NICT), Japan. His research interests include network security, data mining, machine learning, security issues on IPv6, spam analysis, and cryptography theory. He is a member of IEEE.

Masashi Eto received LL.B degree from Keio University in 1999, received the M.E. and Ph.D. degrees from Nara Institute of Science and Technology (NAIST) in 2003, 2005, respectively. From 1999 to 2003, he was a system engineer at Nihon Unisys, Ltd., Japan. He is currently a researcher at National Institute of Information and Communications Technology (NICT), Japan. His research interests include network monitoring, intrusion detection, malware analysis and auto-configuration of the Internetworking. He received the best paper award at the 2007 Symposium on Cryptography and Information Security (SCIS 2007), the best paper award at the 2nd and 3rd Joint Workshop on Information Security (JWIS 2007 and 2008), and the commendation for science and technology by the minister of MEXT, Japan, in 2009.

Daisuke Inoue received his B.E. and M.E. degrees in electrical and computer engineering and Ph.D. degree in engineering from Yokohama National University in 1998, 2000 and 2003, respectively. He joined the Communications Research Laboratory (CRL), Japan, in 2003. The CRL was relaunched as the National Institute of Information and Communications Technology (NICT) in 2004, where he is a senior researcher of the Information Security Research Center. His research interests include security and privacy technologies in wired and wireless networks, incident analysis and response technologies based on network monitoring and malware analysis. He received the best paper award at the 2002 Symposium on Cryptography and Information Security (SCIS 2002), the best paper award at the 2nd and 3rd Joint Workshop on Information Security (JWIS 2007 and 2008), and the commendation for science and technology by the minister of MEXT, Japan, in 2009.

Koji Nakao is the Information Security Fellow in KDDI, Japan. Since joining KDDI in 1979, he has been engaged in the research on multimedia communications, communication protocol, secure communicating system and information security technology for the telecommunications network. He is also an active member of Japan ISMS user group, which was established in the 1st Quarter of 2004. He is the board member of Japan Information Security Audit Association (JASA) and that of Telecom-ISAC Japan, and concurrently, a Technical Group Chairs (ICSS: information communication system security) of the Institute of Electronics, Information and Communication Engineers. He received the B.E. degree of Mathematics from Waseda University, in Japan, in 1979. He received the IPSJ Research Award in 1992, METI Ministry Award and KPMG Security Award in 2006, Contribution Award (Japan ITU), NICT Research Award, Best Paper Award (JWIS) and MIC Bureau Award in 2007. He is a member of IPJS. He has also been a part-time instructor in Waseda University since 2002.