PAPER Special Section on Information and Communication System Security

Finding New Varieties of Malware with the Classification of Network Behavior

Mitsuhiro HATADA[†,††a)] ***and*** **Tatsuya MORI**[†b)]**,** ***Members***

SUMMARY An enormous number of malware samples pose a major threat to our networked society. Antivirus software and intrusion detection systems are widely implemented on the hosts and networks as fundamental countermeasures. However, they may fail to detect evasive malware. Thus, setting a high priority for new varieties of malware is necessary to conduct in-depth analyses and take preventive measures. In this paper, we present a traffic model for malware that can classify network behaviors of malware and identify new varieties of malware. Our model comprises malware-specific features and general traffic features that are extracted from packet traces obtained from a dynamic analysis of the malware. We apply a clustering analysis to generate a classifier and evaluate our proposed model using large-scale live malware samples. The results of our experiment demonstrate the effectiveness of our model in finding new varieties of malware.

key words: *malware communication model, clustering analysis, network behavior classification, new varieties of malware*

1. Introduction

Malware target networked systems that support our society, and a large number of new malware samples have been developed over the last few decades [1]. In addition, another type of malicious software, classified as potentially unwanted programs (PUP), threatens user privacy and computer security [2], [3]. As a countermeasure to malware threats, several technologies aim to detect malware prior to infection, e.g., using virus checkers, blacklisting, and Intrusion Detection System (IDS). However, these pre-infection approaches may fail to detect evasive attacks [4], such as polymorphic malware, various obfuscation techniques, zero-day exploits, and sophisticated social engineering.

One promising complementary approach is to adopt a post-infection approach via malware analyses, which can be roughly classified into static and dynamic analyses. Static analyses can analyze the call flows of malware samples in depth; however, they also slightly increase the burden on the malware analyst. Dynamic analyses [5], [6] execute malware samples on physical or virtual machines. These methods enable us to understand the behaviors of malware quickly with process information, reading/writing files, registry hives, packet traces, memory dumps, and screenshots. We can focus on the packet traces obtained from outside of a host that execute a malware sample without the analysis modules being modified by the malware. In addition, communication with other hosts is indispensable for a recent malware to receive commands, update itself, and continue to attack. Therefore, malware communication logs are valuable because they can be used as an Indicator Of Compromise (IOC), e.g., if a dynamic analyses system reveals communication between a malware-infected client and the command and control server, a Computer Security Incident Response Team should be notified of an IOC. Traffic logs also enable the construction of intrinsic signatures of IDS or rules of Security Information and Event Management that can be used to detect malware activities.

In this paper, we successfully select new varieties of malware from a large number of malware samples under the circumstances that it is difficult to regularly maintain the adequate labeling of malware name. If the network activity of an unknown malware sample is very close to that of a known malware family, it can be treated as a known malware family because a similar malware has already been analyzed. Then, we can set a high priority for the new varieties of malware. Therefore, our goal is to build a novel model of malware traffic based on the results of past measurement studies and several rules of thumb obtained from experts in malware analysis. In addition, we developed a system, illustrated in Fig. 1, that can automatically collect diverse malware samples from various sources, such as honeypots and spamtraps, and analyze the obtained malware with a dynamic analysis system to generate packet traces associated with each malware sample. Then, the system applies the malware communication model to extract feature sets from the packet traces and executes a cluster analysis to generate a classifier.

The contributions of our study can be summarized as follows.

- We develop a generic malware communication model that covers the primary features of communication invoked by various malware families.
- By cluster-analyzing large-scale live malware samples, we confirm that our model can be used to discover malware families that are labeled differently by virus checkers but exhibit similar communication patterns.
- We reveal the effectiveness of our classifier, which can automatically narrow new varieties of malware from 2,019 samples to 649 samples.

Manuscript received September 8, 2016.
Manuscript revised February 6, 2017.
Manuscript publicized May 18, 2017.

[†]The authors are with Waseda University, Tokyo, 169–8555 Japan.
[††]The author is with NTT Communications Corporation, Tokyo, 108–8118 Japan.
a) E-mail: m.hatada@nsl.cs.waseda.ac.jp
b) E-mail: mori@nsl.cs.waseda.ac.jp

DOI: 10.1587/transinf.2016ICP0019

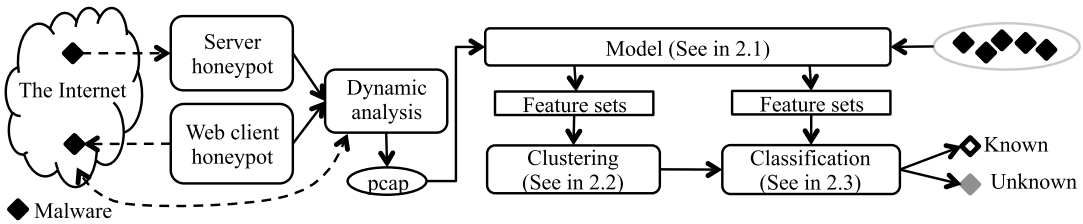

Fig. 1 System overview.

Table 1 Malware-specific features (25 features).

Class	Feature
Activation timing	Time to start communication after binary execution. [sec]
Checking	Number of DNS queries and Number of HTTP requests for major web sites. [18] Number of DNS queries and number of HTTP requests for global IP address check sites.
Information theft	Number of HTTP requests contained inherent host information [10]
C2 (Command and Control)	Number of sessions matched with each 5 public blacklists. Number of IRC sessions over TCP and UDP [19]. Number of P2P sessions over TCP and UDP [20]. Number of sessions with sinkhole hosts [21].
Additional executable files downloading	Number of downloaded executable files.
Spam/scam email sending	Number of DNS queries for MX. Number of SMTP sessions [7], [16], [18].
Probing	Number of ICMP echo requests to the internal host and external hosts [22].
Monetizing	Number of HTTP requests contained advertising words.
Verification	Number of distinct HTTP User-Agent. Maximum and minimum length of HTTP User-Agent [16].

These contributions strongly support security analysts to set a high priority for new varieties of malware and their threats.

The remainder of this paper is organized as follows: Section 2 describes our proposed malware communication model and our approach to clustering and classification. In Sect. 3, we show an overview of the dataset, the validation of our model, and the result of the classification. Section 4 discusses the limitation of our model and our future work; Section 5 summarizes the related studies, and we offer our conclusion in Sect. 6.

2. Methodology

2.1 Malware Communication Model

In this section, we describe a generic model of malware communication patterns. To construct a model, we extract the intrinsic features that concisely represent the behavior of malware communication. When we extract such features, we leverage domain knowledge revealed by prior studies [4], [7]–[16]. While these studies have independently proposed their own features to classify or detect malware, we compiled these features to build a generic model. In addition, we also extracted new features from our own experiences with malware analyses. In total, we extracted 95 features and categorized them into 16 classes, consisting of both malware-specific and generic features. The generic features are helpful to understand the different communication patterns of malware. When we identify new varieties of malware, we can improve this model by adding new feature after a detailed investigation. To extract 95 features from the packet traces, we used TShark [17], which is command-line tool of the strong network analyzer, Wireshark.

2.1.1 Malware-Specific Features

First, we compiled a set of features that represent a typical behavior of malware communication. In general, malware starts to communicate with other hosts soon after being executed or several minutes to hours after the execution. The timing of activation can be arbitrarily set by the malware developers. Once malware is executed on a host, it first checks whether the host is a sandbox or the real victim's host. If it decides that it can continue working, it then launches the rest of its activities, e.g., leaking inherent host information, receiving commands from a remote host, downloading other malicious files, working as a proxy for sending spam email messages, probing other vulnerable hosts, and/or clicking ad-websites for monetization. We translated these activities, which represent the intention of the malware, into a set of features. If a security analyst finds one of these features, he/she can quickly handle the incident. Below, we describe the details of each feature. Table 1 summarizes the features and includes some of the corresponding references.

Timing of activating communication: The timing of activating communication reflects a difference between malware families. Via reverse engineering of malware samples, it has been revealed that some malware developers use APIs such as `Sleep()` or `SleepEx()` to intentionally cause delays to evade dynamic analyses performed over short periods of time. In addition, they can insert unnecessary loop functions to create intentional delays. For example, there is

one malware family that starts communication several minutes after reading several registry keys or files.

Checking network reachability: Another strategy a malware developer may take is to detect whether the malware is running in a controlled environment where the network is not connected to the Internet. If such an environment is detected, the malware will stop running or even destroy itself. To check the network reachability, several malware families attempt to connect to known websites, such as Google or Yahoo [18]. In addition, in our experiences with malware analyses, we have found that malware also make use of websites that are used to check the global IP address of a client. In this paper, we used the Alexa Top 10K sites [23] as a list of major websites. In addition, we manually compiled a list of 10 global IP address checking websites [24]. Note that popular dynamic malware analysis services use these heuristics to detect malware. Therefore, compiling large lists is essential because malware developers may change the websites used to check for network reachability. These features can be extracted by analyzing DNS queries and/or URIs contained in HTTP requests.

Stealing host information: Several malware families have been reported to steal host inherent information and put them into URIs used in HTTP communication [10]. Therefore, if an HTTP request contains inherent host information, it is an intrinsic feature of malware that steals information. Based on our experience, we manually compiled a list of regular expressions that are associated with host inherent information. In particular, we make use of host name, user name, version of Windows OS, MAC address, and display size. When we compile the list, we also check whether a malware accessed to corresponding registry keys. This feature is also seen in PUP, such as adware or pay-per-install [25]. Note that a clear limitation of our approach is that it cannot handle encoded or encrypted strings, which will be addressed in our future studies. This feature can be extracted by analyzing the URIs contained in HTTP requests.

Command and Control (C2): Communicating with C2 servers is one of the most intrinsic features of malware communication. As many previous studies have shown, detecting C2 servers is not an easy task. In this paper, we make use of the following three approaches: a list of C2 servers, sinkhole information, and protocol detection. For the list of C2 servers, we used five known blacklists [26] such as ZeuS Tracker, and DGA [27]. This feature can be detected via the analysis of DNS queries and HTTP requests or destination IP addresses associated with blacklisted hostnames. After a C2 server is taken down, CERTs, police, researchers, or security vendors install a special server using domain names used by the C2 server. These special servers are called sinkhole [21]. The aim of installing a sinkhole server is to let users know about potential malware infection. Sinkhole servers can be detected by monitoring the HTTP response. If an HTTP response contains the "x-sinkhole" header field, it is a sign of a sinkhole server. As previous studies have revealed, IRC [19] and P2P [20] are often used as communication channels between an infected host and C2 servers. To detect these protocols, we make use of the port numbers that each protocol often uses.

Downloading executable files: Downloading executable files is major network activity of malware that are known as Downloader. The objective of Downloader is to download a main binary using a small-sized malware that has only a download function. The number of executable files is identified via the "Content-type" header field in HTTP responses, e.g., application/exe and application/x-msdownload. This header field may not be added, and adequate coding of CGI pages is necessary; therefore, this feature cannot always be precisely extracted.

Sending spam/scam email: Once malware is activated as a spam bot, the malware can send spam emails [7], [16], [18]. First, the malware sends DNS queries for the MX record of major SMTP servers and resolves the IP addresses of the SMTP servers from the DNS responses. Next, the malware attempts to connect to the SMTP servers. If the connection succeeds, the malware starts to send spam emails. Then, the number of the DNS queries for the MX record and the SMTP sessions can easily characterize this behavior.

Probing: To verify the existence of any hosts on the same network or to check the reachability of a specific external host, malware often sends an ICMP echo request [22]. When a host replies to the request, it is assumed that the malware proceeds to the next step. We define the features of probing by the number of internal hosts and external hosts to which the malware sent ICMP echo requests.

Monetizing: So-called Adware is disliked, and users want to block advertising content from being embedded into websites. The risks of Adware are described in Refs. [2], [3], e.g., changing the browser settings, installing malware, and injecting unwanted content. These activities generate revenue for attackers. To detect these activities, Adblock Plus [28] works as a browser plug-in with a number of filters. The number of HTTP requests matched with the Adblock filters is considered to be an effective feature for identifying Adware activities.

Verification: As introduced in Refs. [16], some malware use specific User-Agents with HTTP requests. A malware developer may intend to add a custom User-Agent to verify an infected host. In addition, using varieties of custom User-Agents can evade IDS. The number of distinctive User-Agents and the length of a User-Agent in the HTTP requests are instructive features.

2.1.2 General Features

Regardless of the malware-specific features, there are multiple general features that can specify the traffic patterns. Total traffic size is a very fundamental feature of packet traces. Moreover, TCP, UDP and ICMP are well-known protocols. Encrypted communication such as SSL/TLS is commonly used for legitimate communication to protect user privacy; however, it is also convenient for adversaries to hide messages from network monitoring. Several varieties of mal-

Table 2 General features (70 features).

Class	Feature
All	Total traffic size [bytes]
TCP	Number of sessions, Number of destination hosts, maximum traffic size received/sent per session [bytes], minimum traffic size received/sent per session [bytes]
UDP	Number of sessions, Number of destination hosts, maximum traffic size received/sent per session [bytes], minimum traffic size received/sent per session [bytes]
ICMP	Number of destination hosts for echo request, Number of hosts replied from, Number of unreachable destination hosts
SSL/TLS	Number of sessions, Number of destination hosts, maximum traffic size received/sent per session [bytes], minimum traffic size received/sent per session [bytes]
DNS	Number of queries over TCP, Number of queries over UDP, Number of queries to the external DNS servers, Number of queries with each query type (A, NS, PTR, MX, TXT, AAAA, SRV), Number of responses with each response type (CNAME, SOA)
HTTP	Number of requests with each method (GET, POST, HEAD, M-SEARCH), Patterns of requests (21 patterns; e.g., GET/GET, POST/GET, GET/POST/GET/GET, etc.), Number of responses with each status code (11 types; e.g., 200, 302, 403, etc.)

ware communicate with other hosts by using these protocols. Therefore, basic traffic features, such as the number of destination hosts and the maximum traffic size received/sent per session, are useful to understand the anomalies of network activities.

In addition, there are multiple candidates for useful features with respect to DNS and HTTP because these protocols are usually allowed through a gateway between an organization's network and the Internet. In this paper, we adopt a limited number of the features that were observed via our network monitoring. These general traffic features, which have been the focus of many prior studies [7], [10], [11], [15], [16], [18], [22], are compiled in Table 2.

2.2 Cluster Analysis

We applied a cluster analysis to the feature sets extracted by the malware communication model. Cluster analysis is a fundamental approach to finding patterns without correct labels. Most previous studies have applied supervised learning with labels that were detected via a virus checker. However, because multiple malware samples cannot be detected using virus checkers, it is difficult for security analysts to regularly maintain the adequate labeling. Therefore, we first generated clusters that reflecting the malware communication model.

Prior to the cluster analysis, we filtered out malware samples that had small traffic sizes because the range of Maximum Transmission Unit is between 46 bytes and 1,500 bytes and there is insignificant traffic being dependent on the dynamic analysis environment, e.g., DNS queries for NTP servers and ICMPv6 Router Advertisement. Total traffic sizes less than or equal to 1,000 bytes were not addressed in this study. Standardization was also executed to rescale each feature so that it had a mean of 0 and a standard deviation of 1.

Density-Based Spatial Clustering Applications with Noise (DBSCAN [29] is a clustering algorithm that is often used in data mining. It can find arbitrarily shaped clusters with just two major parameters, ϵ and n. A cluster consists of core points and non-core points that are in areas of high density. A core point has n points within a radius of ϵ and a non-core point is a neighbor of a core point but not a core point itself. $D(x_p, x_q)$ is the distance between the samples x_p and x_q in set of X. A set $N_\epsilon(x_p)$ has samples with greater than n points within a radius of ϵ. To satisfy the following formulas, x_q is directly density-reachable from x_p. Then, a cluster is defined as the maximum set of directory density-reachable samples from an arbitrary sample. We used scikit-learn [30] in Python for the cluster analysis.

$$x_q \in N_\epsilon(x_p)$$
$$|N_\epsilon(x_p)| \geq n$$
$$N_\epsilon(x_p) = \{x_q \in X | D(x_p, x_q) \leq \epsilon\}$$

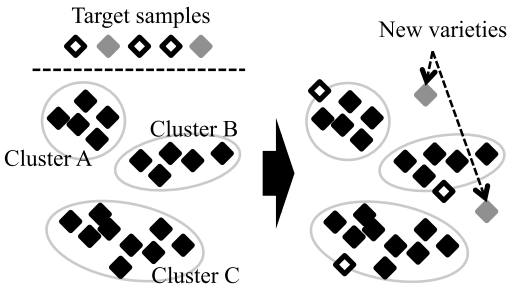

Fig. 2 Classification concept.

2.3 Classification

First, to classify target malware samples into generated clusters or errors, as illustrated in Fig. 2, we extracted the feature set based on the malware communication model described in Sect. 2.1 and conducted the cluster analysis in Sect. 2.2.

Next, we performed the following individual steps on all the target samples that had total traffic volume less than or equal to 1,000 bytes as with the pre-process of cluster analysis. 1) compute the Euclidean distance between a target sample and each sample in all the generated clusters; 2) select the sample in a generated cluster with the minimum distance from the target sample; 3-A) classify the sample into the cluster that the selected sample in step 2 belonged

Table 3 Dataset overview.

	Training data	Testing data
Collection and analysis period	2014-12-01 - 2015-08-31 (9 months)	2015-09-01 - 2015-09-30 (1 month)
Number of distinct samples	21,717	6,078
File type	PE32 (97.9%), Archive (2.0%), Others (0.1%: MsOffice, PE32+(x86-64), MS-DOS)	PE32 (56.1%), Archive (43.6%) Others (0.3%MsOffice, PE32+(x86-64), MS-DOS)
Number of distinct malware name	641	237
Malware type	AdWare (37.0%), HEUR:AdWare (31.7%) Undetectable (16.2%), Downloader (5.7%), Trojan-Downloader (2.4%), Others (7.0%)	Undetectable (33.7%), AdWare (25.0%), Trojan (13.3%), Downloader (10.9%), Trojan-Downloader (5.8%), Others (11.3%)

to if the minimum distance was less than a threshold (θ); 3-B) define the sample as a classification error if the minimum distance is greater than or equal to the θ. Finally, the samples that make it past step 3-B can be identified as new varieties of malware.

3. Experimental Evaluation

In this section, we explain the dataset specification. Then, we evaluate our proposed method with respect to the cluster analysis and classification.

3.1 Dataset

The dataset consists of both training data for the cluster analysis (Sect. 2.2) and testing data for the classification (Sect. 2.3), as shown in Table 3. Both data originate from large-scale live malware samples on the Internet that were collected by two types of high-interaction honeypots, server-type and client-type, from December 2014 to September 2015. The server-type honeypot as described in Ref. [31] works on top of a Windows OS virtual machine that accepts attacks from the Internet. The client-type honeypot [32] works on top of Internet Explorer running on a Windows OS with installed vulnerable plugins. When the client-type honeypot crawls suspicious websites, drive-by download attacks can be observed. We crawled multiple websites that were listed in public blacklists as well as original candidates, such as URLs described in day-to-day e-mail spam. Both honeypots were distributed in several locations countrywide. In order to avoid any damages to our system and other hosts, these honeypots do not allow executing malware and reverting to initial state of a virtual machine. The collected malware were not only executable files but also PDF and Office documents, such as Word, Excel, and PowerPoint.

The collected malware samples were analyzed immediately after collection with a dynamic analysis system that was connected to the Internet. The system starts a guest Windows machine and executes a malware sample within a maximum of 5 min. Then, the system reverts to the initial state of the guest Windows machine. Several access controls are applied to the outgoing traffic that is often used to send spam e-mail and infect other host, e.g., SMTP and NetBIOS, for possible efforts not to cause any damage to other hosts. Dozens of guest Windows machines work simultaneously; however, the packet traces during the analysis are completely separate. If a sample requires a user interaction, such as clicking a confirmation button, the system automatically clicks it by default and proceeds with the malware activity. We removed duplicate malware binaries, sorted them in the order of the first-seen date, and divided them into two periods.

To evaluate the cluster analysis, we built *True categories* based on malware names from a virus checker. We compared the generated clusters with the *True categories*. The number of distinct malware names that the virus checker detected in January 2016 was 641 in the training data and 237 in the testing data. The detection rate was 83.8% in the training data and 66.3% in the testing data. The reason why the virus check was conducted more than 4 months after the collection date is that the accuracy of detection increases with time. We used the full malware names, e.g., Trojan-Downloader.Win32.Adload.icjy, as labels in the *True categories*. Note that 77.4% of undetectable samples were identified as malicious or suspicious by dynamic analysis system.

3.2 Cluster Analysis

3.2.1 Cluster Quality

As in the research field of information retrieval and software similarity [33], we used three evaluation measures, P (Purity), iP (Inverse Purity), and the F-measure, which indicates the harmonic mean between P and iP.

$$P = \frac{1}{N}\sum_{i=1}^{L}\max(n_{i,j})$$

$$iP = \frac{1}{N}\sum_{i=1}^{M}\left(\frac{\sum_{j=1}^{L} n_{i,j}}{\sum_{i=1}^{M} n_{i,j}}\max(n_{i,j})\right)$$

$$F = \frac{1}{\alpha \times \frac{1}{P} + (1-\alpha)\times\frac{1}{iP}} \qquad (0 \le \alpha \le 1)$$

L : the number of generated clusters
M : the number of *True category* clusters
N : the number of samples
$n_{i,j}$: the number of samples that belong to the category j in the cluster i
α : a weight coefficient for P

According to these formulae, a high value of P indicates that many samples of the same virus are named in the

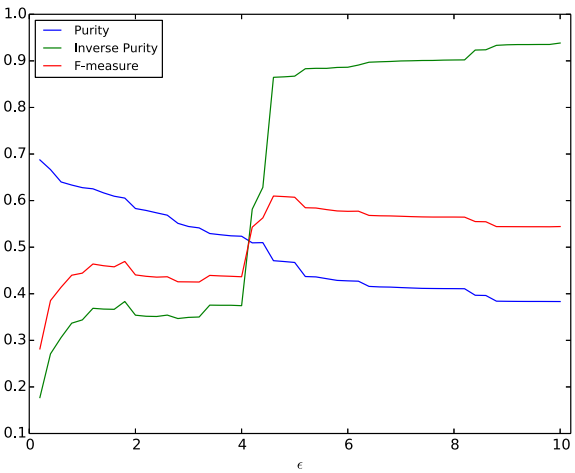

Fig. 3 Evaluation measures with changes of ϵ.

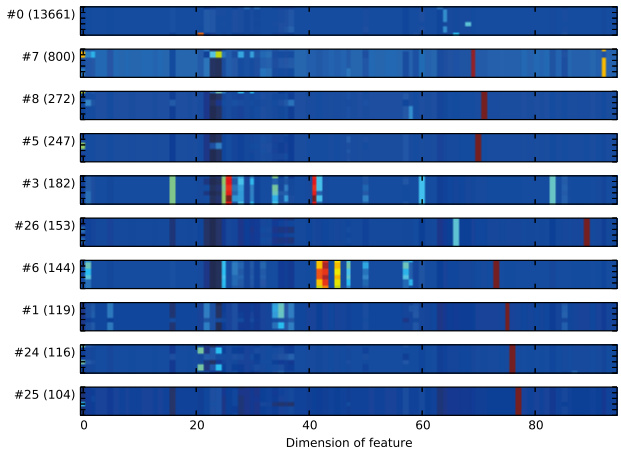

Fig. 4 Visualization of top 10 clusters. The horizontal axis denotes the 95 dimensions of the malware communication features described in Sect. 2.1. The vertical axis indicates 5 samples for each cluster as "#Index of cluster (the number of samples in the cluster)". The dark blue color indicates lower values and the dark red color indicates higher values of each feature.

cluster. A high value of iP indicates that samples of the same virus name center on just one cluster. Then, the value of ϵ with maximum F need to be determined. Note that F depends on ϵ.

$$\widehat{\epsilon} = \arg\max_{\epsilon} F(\epsilon)$$

In this study, we set $\alpha = 0.5$. In addition, we set another parameter of DBSCAN to $n = 1$. It would be useful if we could collect several samples for each cluster, e.g., at least five samples per cluster, etc. However, in practice, it is not an easy task to collect the fixed quantity of malware that has similar network activities. Therefore we decided to make a choice that we fully make use of the information available by generating a cluster even though it consists of one sample. We note that this choice is useful especially for finding new varieties of malware.

A preliminary experiment indicated that the total traffic size for approximately 21% of the samples in the training data was less than 1,000 bytes. Therefore, we used 17,167 samples for this evaluation. Figure 3 shows the changes in the three evaluation measures when ϵ changed from 0.2 to 10. The result shows that the highest value of F is 0.61 when ϵ is 4.6. Given the fact that we used a limited scope of features, i.e., only network traces, this is a reasonable quantity compared to other research fields, e.g., $\max(F) = 0.98$ in software classification [33] and $\max(F) = 0.79$ in information retrieval [34]. In fact, Ref. [33] achieves this result by integrating source code clustering and specification document clustering. In this paper, we used only network traces and the full malware names as labels in the *True categories*. Although we believe that F can be improved by various approaches such as integration of other analysis features, the aim of this work was to demonstrate the effectiveness of generated clusters obtained through the analysis of network traces. In this case, the cluster analysis, including pre-processing, was performed in 83.1 s with a Mac OS X (CPU: 1.7 GHz Intel Core i7, Memory: 8GB 1600MHz DDR3). Note that 310 of 439 clusters had one sample in one cluster. Of these, 118 clusters could not be detected by the virus checker. Of the 439 clusters extracted, 55 had multiple malware names. We summarize the breakdown of the obtained statistics and the examples of malware names in Table A· 1. Naturally, undetectable samples were ignored when computing the three evaluation measures.

3.2.2 Cluster Visualization

Figure 4 visualizes the top 10 clusters to more clearly illustrate the malware communication pattern. It appears that each cluster is well characterized by the malware communication model. We give three examples of typical network activities below.

Cluster 0 is the most popular cluster, and it involves 308 distinct malware names from the 13,661 samples, and AdWare.Win32.MultiPlug.heur accounts for 43% of the samples. *Checking network reachability* and *downloading executable file* behaviors were observed. Four HTTP GET requests and two HTTP POST requests were often sent; however samples lacked distinguishing characteristics.

AdWare.Win32.iBryte.jif composes approximately 60% of Cluster 7. *Timing of activating communication* of samples in this cluster was 30 s after execution, two different lengths of User-Agent in the HTTP requests, and they received one HTTP response with a status code of 410.

Cluster 3 contained 182 samples of Trojan-Downloader. Win32.Adload.icjy. *Checking network reachability* and *downloading executable file* were conducted, as with Cluster 0; however, a number of SSL/TLS sessions were also found.

3.3 Classification

As a result of the cluster analysis, samples with different malware names were gathered into clusters according to common network activities. Conversely, samples with the

Table 4 Appearance ratio of true category labels for classification results.

(a) θ=3.0

	R=0.7	R=0.5	R=0.25	R=0.2
Q=1	0.773	0.773	0.769	0.559
Q=3	0.773	0.773	0.769	0.559
Q=5	0.773	0.773	0.769	0.559
Q=7	0.773	0.773	0.769	0.559

(b) θ=4.6

	R=0.7	R=0.5	R=0.25	R=0.2
Q=1	0.742	0.731	0.726	0.595
Q=3	0.743	0.732	0.727	0.595
Q=5	0.747	0.736	0.731	0.598
Q=7	0.747	0.737	0.732	0.599

(c) θ=7.0

	R=0.7	R=0.5	R=0.25	R=0.2
Q=1	0.586	0.579	0.568	0.505
Q=3	0.599	0.592	0.581	0.516
Q=5	0.602	0.595	0.584	0.519
Q=7	0.602	0.595	0.584	0.519

Table 5 Malware names included both in classified samples and unclassified samples.

AdWare.JS.MultiPlug.p	AdWare.NSIS.Adload.i
AdWare.Win32.AdLoad.flrs	AdWare.Win32.Agent.izjh
AdWare.Win32.Amonetize.bgnd	AdWare.Win32.Amonetize.bhvk
AdWare.Win32.Eorezo.zby	AdWare.Win32.MegaSearch.am
Downloader.NSIS.OutBrowse.bm	HEUR:AdWare.Win32.Generic
HEUR:Trojan.Win32.Generic	RiskTool.Win32.StartPage.bg
Trojan-Downloader.Win32.Adload.icjy	Trojan-Dropper.Win32.Agent.peok
Trojan-Dropper.Win32.Injector.hcun	Trojan-Dropper.Win32.Necurs.drf
Trojan.Win32.Agent.ifbi	Virus.Win32.Parite.b
WebToolbar.Win32.SearchSuite.ae	

Table 6 Result of new varieties of malware narrowed by proposal method. (θ=3.0)

	Number of samples	Number of distinct malware names
Undetectable	1,661	n/a
Detectable	2,019	165
Classified	472	64
in the testing data	106	44
Unclassified	1,547	120
in the testing data	649	90

same malware names also belonged to different clusters. To evaluate the classification, we did not specify a representative of the malware names in each cluster; rather, we allowed representatives of multiple malware names in each cluster to maintain flexibility. In particular, a cluster should have a minimum number of samples Q; then, we can compute the appearance ratio of the top R% of malware names in a cluster for the classification results. 2,398 samples in the testing data are small size traffic so that these samples are not subject to classify. The results are shown in Table 4 for three different threshold distances (θ = 3.0, 4.6, and 7.0) when judging an error. Following the preliminary experiment results, samples in the testing data were able to be classified 0%, 19.2%, and 90.5% when θ was 2.0, 3.0, and 7.0. 4.6 is same value to the parameter ϵ of DBSCAN. When R is greater than 0.5, the best value of the appearance ratio becomes the value previously shown value. Therefore, we omit to present all the results and show representative results in Table 4. From Table 4, we see that even though a higher θ increases the number of misclassified samples, we obtain a maximum appearance ratio 77.3% when $\theta = 3.0$ and $R \geqq 0.5$. Under these conditions, it took only 0.29 s to classify a sample. The false cases were 22.7% of samples that did not appear in the top 50% of malware names in the nearest cluster. The primary cause was that samples with the same malware names were classified into multiple clusters or unclassified. For example, the number of same malware names included both in classified samples and unclassified samples was 19 as listed in Table 5. These malware samples are roughly classified into Adware and Downloader/Dropper, and usually download other content such as advertising data or other malicious files. The content can be changed with the timing of dynamic analysis; therefore, the results of classification could become different even if the sample that had the same malware name.

To evaluate the effectiveness of the proposed classification to find new varieties of malware, we counted the number of classification error samples seen only in the testing data (Table 6). Because the testing data was collected after the collection period of the training data, samples with malware names seen only in the testing data were considered to be new varieties. The results demonstrate that 649 samples (90 distinct malware names) were revealed as new varieties of malware by our proposed method. The number of samples seen only in the testing data where classification succeeded was 106 (44 distinct malware names). These samples, which could be potentially overlooked, are new varieties; however, all samples were classified into Cluster 0, as explained in Sect. 3.2.2, and their distinguishing characteristics were not observed. Therefore, we succeeded in narrowing the malware samples from 2,019 to 649 of new varieties.

4. Discussion

The effectiveness of our approach was demonstrated via an experimental evaluation. However, there are limitations to the malware collection and analysis, the malware communication model, the cluster analysis, and the classification with respect to their inherent challenges.

Malware without network behavior: The behavior of recent malware depends on its network interactions, as described in Sect. 2.1. For example, the *DarkS eoul* [35] malware only overwrites the Master Boot Record of the computer, instructing it to fail booting. Consequently, it does not need to communicate with other hosts to achieve this specific objective. In fact, numerous malware without network behavior exist, and this is a severe limitation of our model.

Evasion malware: Evasion malware constitutes a common weakness in dynamic analysis systems. There are several techniques to avoid detection during an analysis. In a run-

time environment, checking for a debugger's presence, fingerprinting a virtual machine, checking for Internet reachability, and checking for a global IP address are popular techniques to avoid being analyzed [36]. In other cases, malware may require user interaction, run actively after being rebooted, or run only on a specific date. Our system includes several techniques to counteract these types of malware methods; however, these techniques are not yet sufficient.

Malware coverage: At present, our system partially covers many malware infection paths to a host. Malware as an email attachment, however, remains a significant threat for enterprises. Moreover, malware can infect other platforms, such Mac OS and Linux. We cannot identify the traffic patterns in these malware communications because our system is based on Windows OS. Notwithstanding the cost, adding other sources of malware and dynamic analysis systems can ideally solve this issue. However, it is practically impossible to cover all malware; therefore, one approach is to focus on the local data associated with protecting specific objects, such as a company, a business domain, or an ISP. However, the generalized model presented here can incorporate data on malware from multiple sources and environments.

In this paper, the testing data also partially covers all features of malware. If known malware samples that have never been analyzed are included in the testing data, our method identifies them as new malware at that point. It is necessary for security analysts to judge them malware and update the clusters regularly. There is another way to update the clusters. If malware samples in the testing data are detected by antivirus software with a high reputation, these samples can be included in the training data. Relearning process of clusters is crucial to improving our system.

Model Improvements: As described in Sect. 3.2, our proposed model successfully identifies the traffic patterns in malware communications. In terms of malware-specific features, the classes of features that are entirely covered are based on the initial malware behavior; however, each feature can be expanded by both increasing the matching conditions and upgrading the matching capabilities. There are multiple public and continuously upgraded data sources that can be used to increase the matching conditions, such as major sites for checking the Internet reachability, sites for checking global IP addresses, and C2 blacklists. Inherent host information can be used for this purpose as well, even though it depends on the dynamic analysis environment; this information can be added by analyzing the host internal activity. Malware also attacks other hosts by sending exploit codes and massive traffic; these attacking behaviors can be defined by another feature while maintaining the signature associated with the attacks.

Upgrading the model's matching capabilities is also important. Malware can retrieve and send information, download executable files, and communicate with C2 servers with encoding or encryption. Detecting and analyzing encrypted malware communication [37] is a continuous challenge.

By contrast, general traffic patterns can contribute to the above challenges. Trends/changes in general malware features require investigating malware communication methods with other hosts. Once this new behavior is identified based on a more in-depth analysis, the new feature can be integrated into the model.

Clustering and Classification: In this paper, the labels of the *True categories* were added using virus checker. However, there were many undetectable samples in the dataset. We did not evaluate these undetectable samples; however, it is possible to label samples via the results of static and/or dynamic analyses. We could also choose a better algorithm for the cluster analysis via comparisons and optimize the parameters of the algorithm. In terms of classification, we did not specify a representative of the malware name in each cluster. Representatives of multiple malware names in each cluster were allowed for flexibility, and the Euclidean distance was simply used to classify a sample into a cluster. It is necessary to improve these approaches for a more precise determination of a new threat. Moreover, the continuous maintenance of the generated clusters should be addressed in a future study.

5. Related Studies

Many previous studies have attempted to achieve automated malware classification and detection by applying supervised learning with labels that are detected using a virus checker. However, it is not an easy task to regularly maintain adequate labels because multiple malware samples cannot be detected using a virus checker. Our study is a first approach for security analysts to find new varieties of malware. Here, we review previous studies according to malware classification, malware detection, network signature generation, and malicious network behavior detection.

Malware classification. : In Ref. [18], an efficient malware classification system was presented that based on the protocol-aware and state-space features with potential input for various classification methods. To extract fine-grained information, protocol-aware modeling, such as HTTP and SMTP messages, is useful. The objective of state-space modeling is to deal with network behavior of unknown protocols. Ref. [18] evaluated several classification algorithms against 6,000 unique and active malware samples belonging to 20 families. In Ref. [7], the authors defined behavior graphs based on extracted network flows from packet traces. To generate the graphs, they used dependencies between network flows. In their graphs, when the IP address in the DNS response was associated with the destination IP address of the HTTP communication, an edge was created from the DNS node and HTTP node. Once the graph was generated, 10 features could be extracted to classify the malware samples, such as the number of nodes and the number of direct nodes that form the root node. A J48 decision tree was performed on a few thousand labeled samples from 13 malware families.

Malware detection. : BotFinder [8] was able to detect

malware infections via a traffic pattern analysis without IP blacklisting or deep packet inspection. The authors applied the non-supervised training of six known bot families using the average time interval between the start times of two subsequent flows in the trace, the average duration of connections, and the average number of source bytes and destination bytes per flow. Zhang, et al. [10] introduced a new traffic analysis method to detect stealthy malware activities on a host by discovering the underlying triggering relations of huge network events. The experimental results of focusing on DNS and HTTP events showed its effectiveness against malicious browser extension, data exfiltration malware, and DNS tunneling bots. Bartos et al. [38] proposed a robust representation to classify evolving malware behaviors. The classifier uses a statistical feature of the bags, which are sets of network flows. The proposed classification was applied to proxy logs (only HTTP flows) on large corporate networks, detected 2,090 new bags of malware variants, and verified 9 of 10 alerts as malicious.

Network signature generation. : Perdisci et al. [15] presented an HTTP-based malware clustering and signature-generating system to find simple statistical similarities in HTTP traffic generated by different malware samples, the number of HTTP requests, GET requests, POST requests, and the average URL length. Besides the above features, the structure of each HTTP request with the method, page, parameter names, and parameter values was used assume similarities in server-side applications. Their experimental results on over 25,000 malware samples showed their system was effective at clustering and generating signatures. FIRMA [16] achieved malware clustering and network signature generation. To cluster traffic, the authors extracted the source and destination of the port number and IP address, the traffic size, and the hostname of each HTTP, IRC, SMPT, TCP, and UDP request. In addition, a message tree was generated following the method in HTTP and the command in IRC and SMTP. Based on the clustering, a set of network signatures for each malware family was produced.

Malicious network behavior detection. : There have also been interesting studies to detect malicious network behavior that have not focused on only malware activities. In Ref. [12], approximately 40 features of flow level without deep packet inspection were used to discriminate varieties of malicious network traffic. To distinguish normal and anomalous computer behavior, a one-class classifier based on feed-forward neural network was proposed in Ref. [13]. The incoming and outgoing TCP, UDP and HTTP packets sizes of each request per second were major parameters. Processor usage, memory usage, and disk input/output were also used as metrics. The results of experiments on TCP/UDP flooding to web servers showed a promising capacity for fast anomaly detection. SNAPS [14] is a coherent system that can discove network anomalies for domain experts by adapting deep packet inspection, machine learning, and visualization. MAGMA [39] is a classifier to identify malicious network activity based on a multi-layer network connectivity graph.

6. Conclusions

In this paper, we proposed a novel classifier to find new varieties of malware using packet traces generated by the malware. We built a malware communication model that covered the primary features of communication invoked by various malware families. The model consists of 25 malware-specific features and 70 general traffic features that leverage previous studies and several rules of thumb obtained from experts. Using multiple types of features can improve robustness against attacks that hide individual activity. We verified that our model helped discover different malware families with similar communication patterns via the evaluation of cluster analysis using large-scale live malware samples. We also revealed the value of the clustering parameter with maximum F-measure and visualized clusters with explanations of representative network activity. The experimental results demonstrated that the classification accuracy was 77.3%, and the malware samples were narrowed from 2,019 to 649 new varieties. We assume that the cluster should be updated after manually judging the maliciousness of samples extracted by our method. During this manual inspection process, dynamic analysis can keep working. We also note that dynamic analysis process can be scaled-out by adopting the parallelization approach. Therefore, we focused on the analysis of execution time for clustering and classification processes. We were able to conduct the cluster analysis of the 21,717 malware samples in 83.1 s and were able to classify a sample in 0.29 s. The limitations of this study and our future work were discussed from the various viewpoints of malware without network behavior, evasive malware, malware coverage, model improvement, and clustering and classification methods. These results and discussions will support security analysts in prioritizing the vast number of new malware threats.

Acknowledgments

A part of this work was supported by JSPS Grant-in-Aid for Scientific Research B, Grant Number JP16H02832.

References

[1] "Malware Statistics & Trends Report | AVTest Institute." https://www.av-test.org/en/statistics/malware/.
[2] "Threat intelligence report - Unwanted software." https://www.microsoft.com/security/portal/enterprise/threatreports_october_2015.aspx.
[3] P. Kotzias, L. Bilge, and J. Caballero, "Measuring pup prevalence and pup distribution through pay-per-install services," Proc. 25th USENIX Security Symposium, USENIX Security '16, pp.739–756, Aug. 2016.
[4] M. Bailey, J. Oberheide, J. Andersen, Z.M. Mao, F. Jahanian, and J. Nazario, "Automated classification and analysis of internet malware," Proc. 10th International Conference on Recent Advances in Intrusion Detection, RAID '07, pp.178–197, Sept. 2007.
[5] "Cuckoo sandbox." http://www.cuckoosandbox.org/.
[6] "Malwr." https://malwr.com/.

[7] S. Nari and A.A. Ghorbani, "Automated Malware Classification based on Network Behavior," Proc. International Conference on Computing, Networking and Communications 2013, ICNC '13, pp.642–647, Jan. 2013.
[8] F. Tegeler, X. Fu, G. Vigna, and C. Kruegel, "BotFinder: Finding bots in network traffic without deep packet inspection," Proc. 8th International Conference on Emerging Networking Experiments and Technologies, CoNEXT '12, pp.349–360, Dec. 2012.
[9] L. Invernizzi, S. Miskovic, R. Torres, S. Saha, S.J. Lee, C. Kruegel, and G. Vigna, "Nazca: Detecting malware distribution in large-scale networks," Proc. ISOC Network and Distributed System Security Symposium, NDSS '14, pp.1–16, Feb. 2014.
[10] H. Zhang, D.D. Yao, and N. Ramakrishnan, "Detection of stealthy malware activities with traffic causality and scalable triggering relation discovery," Proc. 9th ACM Symposium on Information, Computer and Communications Security, ASIACCS '14, pp.39–50, Jun. 2014.
[11] H. Mekky, A. Mohaisen, and Z.-L. Zhang, "POSTER: Blind Separation of Benign and Malicious Events to Enable Accurate Malware Family Classification," Proc. 2014 ACM Conference on Computer and Communications Security, CCS '14, pp.1478–1480, Nov. 2014.
[12] J.M. Beaver, C.T. Symons, and R.E. Gillen, "A learning system for discriminating variants of malicious network traffic," Proc. Eighth Annual Cyber Security and Information Intelligence Research Workshop, CSIIRW '13, pp.23:1–23:4, Jan. 2013.
[13] V. Eliseev and Y. Shabalin, "Dynamic response recognition by neural network to detect network host anomaly activity," Proc. 8th International Conference on Security of Information and Networks, SIN '15, pp.246–249, Sept. 2015.
[14] B. Cappers and J.J. van Wijk, "SNAPS: Semantic network traffic analysis through projection and selection," Proc. 2015 IEEE Symposium on Visualization for Cyber Security, VizSec '15, pp.1–8, Oct. 2015.
[15] R. Perdisci, W. Lee, and N. Feamster, "Behavioral clustering of http-based malware and signature generation using malicious network traces," Proc. 7th USENIX Conference on Networked Systems Design and Implementation, NSDI '10, pp.1–14, April 2010.
[16] M.Z. Rafique and J. Caballero, "FIRMA: Malware Clustering and Network Signature Generation with Mixed Network Behaviors," Proc. 16th International Symposium on Research in Attacks, Intrusions and Defenses, RAID '13, pp.144–163, Oct. 2013.
[17] "Tshark." https://www.wireshark.org/docs/man-pages/tshark.html.
[18] M.Z. Rafique, P. Chen, C. Huygens, and W. Joosen, "Evolutionary algorithms for classification of malware families through different network behaviors," Proc. 2014 Annual Conference on Genetic and Evolutionary Computation, GECCO '14, pp.1167–1174, July 2014.
[19] G. Gu, J. Zhang, and W. Lee, "Botsniffer: Detecting botnet command and control channels in network traffic," NDSS '08, pp.1–18, Feb. 2008.
[20] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection," Proc. 17th Conference on Security Symposium, SS '08, pp.139–154, July 2008.
[21] S. Shin and G. Gu, "Conficker and beyond: A large-scale empirical study," Proc. 26th Annual Computer Security Applications Conference, ACSAC '10, pp.151–160, Dec. 2010.
[22] J.A. Morales, A. Al-Bataineh, S. Xu, and R. Sandhu, "Analyzing and exploiting network behaviors of malware," in Security and Privacy in Communication Networks, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol.50, pp.20–34, 2010.
[23] "Alexa Top Sites." http://www.alexa.com/topsites.
[24] "WhatIsMyIP.com." https://www.whatismyip.com/.
[25] "Pay-Per-Install: The New Malware Distribution Network." https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/pay_per_install.pdf.
[26] "abuse.ch." https://www.abuse.ch/.
[27] "Domain feed of known DGA domains." http://osint.bambenekconsulting.com/feeds/dga-feed.txt.
[28] "Adblock plus." https://easylist-downloads.adblockplus.org/easylist.txt.
[29] M. Ester, H. peter Kriegel, J.S., and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," Proc. 2nd International Conference on Knowledge Discovery and Data Mining, KDD '96, pp.226–231, Aug. 1996.
[30] "scikit-learn." http://scikit-learn.org/.
[31] K. Aoki, Y. Kawakoya, M. Akiyama, M. Iwamura, T. Hariu, and M. Ito, "Investigation and Understanding Active/Passive Attacks," IPSJ Journal, vol.50, no.9, pp.2147–2162, Sept. 2009. (in Japanese).
[32] M. Akiyama, K. Aoki, Y. Kawakoya, M. Iwamura, and M. Ito, "Design and Implementation of High Interaction Client Honeypot for Drive-by-download Attacks," IEICE Trans. on Commun., vol.E93-B, no.5, pp.1131–1139, May 2010.
[33] Y. Kishimoto, A. Sakamoto, and T. Kobayashi, "Extracting of Derived Relations among Software Products Using Both Source Cord and Specification Document," IPSJ Journal, vol.53, no.3, pp.1137–1149, March 2012. (in Japanese).
[34] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo, "A comparison of extrinsic clustering evaluation metrics based on formal constraints," Inform. Retrieval, vol.12, no.4, pp.461–486, 2009.
[35] "DarkSeoul: SophosLabs identifies malware used in South Korean internet attack." https://nakedsecurity.sophos.com/2013/03/20/south-korea-cyber-attack/.
[36] D. Kirat, G. Vigna, and C. Kruegel, "Barecloud: Bare-metal analysis-based evasive malware detection," Proc. 23rd USENIX Conference on Security Symposium, USENIX Security '14, pp.287–301, Aug. 2014.
[37] H. Zhang, C. Papadopoulos, and D. Massey, "Detecting encrypted botnet traffic," Proc. 2013 IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS '13, pp.163–168, April 2013.
[38] K. Bartos, M. Sofka, and V. Franc, "Optimized invariant representation of network traffic for detecting unseen malware variants," Proc. 25th USENIX Security Symposium, USENIX Security '16, pp.807–822, Aug. 2016.
[39] E. Bocchi, L. Grimaudo, M. Mellia, E. Baralis, S. Saha, S. Miskovic, G. Modelo-Howard, and S.-J. Lee, "MAGMA: Network Behavior Classifier for Malware Traffic," Computer Networks "Special issue on Traffic and Performance in the Big Data Era," vol.109, pp.142–156, April 2016.

Appendix A:

Table A· 1 Number of clusters with multiple malware names and examples within top 10.

Number of malware names	Number of clusters	Example of cluster index	Malware names (Number of distinct samples)	
2	34	4	Trojan-Downloader.Win32.Adload.icjy (28)	Trojan-Dropper.Win32.Agent.sbcy (2)
3	4	32	HEUR:AdWare.Win32.MultiPlug.heur (20)	AdWare.Win32.MultiPlug.ofcb (6)
			Trojan-Banker.Win32.Agent.upd (1)	
4	3	1	AdWare.Win32.iBryte.jif (85)	AdWare.Win32.iBryte.joo (22)
			HEUR:AdWare.Win32.MultiPlug.heur (4)	HEUR:AdWare.Win32.Generic (2)
6	4	8	AdWare.Win32.MultiPlug.bwof (229)	AdWare.Win32.MultiPlug.oaul (16)
			AdWare.Win32.Agent.iziy (5)	HEUR:AdWare.Win32.MultiPlug.heur (2)
			HEUR:AdWare.Win32.Generic (1)	Downloader.Win32.DownloAdmin.fgb (1)
7	2	5	AdWare.Win32.MultiPlug.bwof (197)	AdWare.Win32.MultiPlug.oaul (28)
			HEUR:AdWare.Win32.MultiPlug.heur (3)	AdWare.Win32.Agent.iziy (3)
			WebToolbar.Win32.Neobar.a (1)	AdWare.Win32.MultiPlug.oarp (1)
			AdWare.Win32.Amonetize.xhj(1)	
8	2	2	Downloader.Win32.DriverUpd.ipb (14)	AdWare.Win32.Agent.djcr (12)
			AdWare.Win32.SoftPulse.p (8)	Trojan.Win32.Inject.untk (5)
			Downloader.Win32.DriverUpd.hlh (2)	Downloader.Win32.Agent.cdvd (1)
			AdWare.Win32.SoftPulse.ybk (1)	AdWare.Win32.SoftPulse.beud (1)
14	1	24	AdWare.Win32.MultiPlug.oaqn (36)	WebToolbar.Win32.SearchSuite.ae (33)
			HEUR:Trojan.Win32.Generic (5)	HEUR:RiskTool.Win32.Generic (2)
			HEUR:AdWare.Win32.MultiPlug.heur (1)	HEUR:AdWare.Win32.Lollipop.heur (1)
			HEUR:AdWare.Win32.Generic (1)	AdWare.Win32.MultiPlug.ofcb (1)
			AdWare.Win32.Kraddare.aip (1)	AdWare.Win32.AdLoad.flxq (1)
16	1	(7	AdWare.Win32.iBryte.jif (483)	AdWare.Win32.iBryte.jke (96)
			AdWare.Win32.iBryte.jhg (53)	AdWare.Win32.iBryte.jkb (24)
			AdWare.Win32.MultiPlug.oaul (22)	AdWare.Win32.iBryte.jka (18)
			AdWare.Win32.iBryte.jjo (15)	HEUR:Trojan.Win32.Generic (15)
			AdWare.Win32.Agent.iziy (9)	HEUR:AdWare.Win32.MultiPlug.heur (3)
19	1	12	AdWare.Win32.MultiPlug.oaul (27)	AdWare.Win32.iBryte.jjy (8)
			AdWare.Win32.iBryte.jhg (6)	AdWare.Win32.MultiPlug.oaqn (3)
			AdWare.JS.MultiPlug.p (3)	HEUR:Downloader.Win32.Generic (2)
			HEUR:AdWare.Win32.OutBrowse.heur (1)	HEUR:AdWare.Win32.Generic (1)
			HEUR:AdWare.NSIS.Agent.heur (1)	AdWare.Win32.SoftPulse.p (1)
22	1	18	Downloader.Win32.LMN.afd(3)	AdWare.Win32.SoftPulse.p (2)
			AdWare.Win32.Agent.djcr (2)	WebToolbar.Win32.Agent.azm (1)
			WebToolbar.Win32.Agent.avl (1)	RemoteAdmin.Win32.Ammyy.xcs (1)
			RemoteAdmin.Win32.Ammyy.jlu (1)	RemoteAdmin.Win32.Ammyy.hq (1)
			RemoteAdmin.Win32.Ammyy.ani (1)	RemoteAdmin.Win32.Ammyy.and (1)
26	1	20	AdWare.Win32.AdLoad.flxq (26)	HEUR:Trojan.Win32.Generic (13)
			Downloader.Win32.Agent.ajuq (8)	AdWare.NSIS.Adload.i (5)
			WebToolbar.Win32.SearchSuite.ae (4)	AdWare.Win32.Kraddare.aip (3)
			Trojan-Dropper.Win32.Injector.hxbu (3)	HEUR:Downloader.Win32.Generic (1)
			HEUR:Downloader.NSIS.SoftBase.gen (1)	HEUR:AdWare.Win32.SoftPulse.heur(1)
308	1	0	HEUR:AdWare.Win32.MultiPlug.heur (6002)	AdWare.Win32.MultiPlug.ofcb (1462)
			AdWare.Win32.MultiPlug.oaul (1100)	AdWare.Win32.iBryte.jjy (412)
			AdWare.Win32.iBryte.jhg (370)	Downloader.Win32.Somato.g (308)
			HEUR:AdWare.Win32.OutBrowse.heur (228)	Downloader.Win32.Morstar.bad (207)
			Downloader.Win32.Agent.dlzx (204)	Downloader.Win32.Agent.clgu (183)

Mitsuhiro Hatada was born in 1978. He is currently a Ph.D. student with particular interest in anti-malware. He received his B.E. and M.E. degrees in computer science and engineering from Waseda University in 2001 and 2003, respectively. He joined NTT Communications Corporation in 2003 and has been engaged in the R&D of network security and anti-malware. He is a member of IEICE and IPSJ.

Tatsuya Mori is currently an associate professor at Waseda University, Tokyo, Japan. He received B.E. and M.E. degrees in applied physics, and Ph.D. degree in information science from the Waseda University, in 1997, 1999 and 2005, respectively. He joined NTT lab in 1999. Since then, he has been engaged in the research of measurement and analysis of networks and cyber security. From Mar 2007 to Mar 2008, he was a visiting researcher at the University of Wisconsin-Madison. He received Telecom System Technology Award from TAF in 2010 and Best Paper Awards from IEICE and IEEE/ACM COMSNETS in 2009 and 2010, respectively. Dr. Mori is a member of ACM, IEEE, IEICE, and USENIX.