IEICE TRANSACTIONS

on Communications

VOL. E103-B NO. 1
JANUARY 2020

A PUBLICATION OF THE COMMUNICATIONS SOCIETY

The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3chome, Minato-ku, TOKYO, 105-0011 JAPAN

PAPER *Special Section on Internet Architecture, Applications and Operation Technologies for a Cyber-Physical System*

IoT Malware Analysis and New Pattern Discovery Through Sequence Analysis Using Meta-Feature Information

Chun-Jung WU[†a)], **Shin-Ying HUANG**[††], ***Nonmembers***, **Katsunari YOSHIOKA**[†,†††], ***and*** **Tsutomu MATSUMOTO**[†,†††], ***Members***

SUMMARY A drastic increase in cyberattacks targeting Internet of Things (IoT) devices using telnet protocols has been observed. IoT malware continues to evolve, and the diversity of OS and environments increases the difficulty of executing malware samples in an observation setting. To address this problem, we sought to develop an alternative means of investigation by using the telnet logs of IoT honeypots and analyzing malware without executing it. In this paper, we present a malware classification method based on malware binaries, command sequences, and meta-features. We employ both unsupervised or supervised learning algorithms and text-mining algorithms for handling unstructured data. Clustering analysis is applied for finding malware family members and revealing their inherent features for better explanation. First, the malware binaries are grouped using similarity analysis. Then, we extract key patterns of interaction behavior using an N-gram model. We also train a multiclass classifier to identify IoT malware categories based on common infection behavior. For misclassified subclasses, second-stage sub-training is performed using a file meta-feature. Our results demonstrate 96.70% accuracy, with high precision and recall. The clustering results reveal variant attack vectors and one denial of service (DoS) attack that used pure Linux commands.

key words: *IoT malware, botnet, denial of service, text mining, N-gram, classification, clustering*

1. Introduction

Internet of Things (IoT) is a network of physical devices, such as vehicles, furniture, and buildings, that are embedded with electronics, sensors, and networking abilities. Connectivity enables these objects to collect and exchange data for further application. However, in October 2016, the IoT malware called Mirai executed the largest-ever distributed denial of service (DDoS) attack against Dyn DNS in 2016, enlisting approximately 100,000 Mirai IoT botnet nodes for a reported attack rate of up to 1.2 Tbps [1]. According to the report from Kaspersky in Sept. 2018, Mirai is still the most popular IoT malware family for cybercriminals (20.9%). Moreover, the most popular attack and infection vector against devices remains the telnet service (75.4%) [2]. Although signature-based detection methods are sensitive to the structures of existing malware samples, even a small change in a malware program could alter its signature sufficiently to thwart antivirus detection. Therefore, an urgent necessity is to analyze IoT malware and related logs to recognize the behavior of unfamiliar threats and thus assist organizations in mounting a timely and appropriate defense.

Gartner estimated that 6.4 billion IoT devices were in use in 2016, and this number is projected to grow to 20.8 billion by 2020 [3]. Embedding information and communication technology into devices thus represents an ongoing trend. The nature of IoT presents challenges in establishing a comprehensive security mechanism. These challenges arising from IoT devices are as follows:

1. Most IoT devices are always online.
2. Most utilize simple, low-level hardware.
3. IoT devices have a variety of CPU architectures and OS.
4. Antivirus and monitoring services are lacking.
5. Diverse developers result in a lack of unified standards.
6. IoT malware attack patterns are continually evolving, with an extremely large damage scope.

Few of them are protected by antivirus software. To analyze the threat of IoT malware, Pa et al. [4] proposed IoTPOT, a honeypot that observes cyberattacks against IoT devices, focusing on telnet-based attacks; it emulated IoT devices that accepting telnet protocol connections. When attackers access IoTPOT, it records the entire netflow and maintains logs for further analysis, such as downloading samples. Since 2015, 6,016,030 download attempts from 1,085,664 different hosts have been successfully observed and over 40,000 malware samples downloaded. Moreover, 124,517,838 telnet session logs have been collected, recording all the shell command input sent by the attackers. IoTPOT thus represents a useful method of collecting samples, analyzing threat behavior, and understanding IoT cyberattacks. However, the enormous data size resulted in huge time and resource cost when analyzing their patterns and relationship. It is an urgent necessity to create an appropriate view which analyzes the incoming data in depth and utilize our resource efficiently.

The purpose of this study is to apply machine learning techniques to create a simplified and accurate view of IoT cyberattacks. The method determines categories of malware by analyzing its meta-features and command sequences. Its contributions may be summarized as follows:

1. We proved that similar IoT malware binaries conduct

Manuscript received February 15, 2019.
Manuscript revised June 3, 2019.
Manuscript publicized August 5, 2019.

[†] The authors are with the Graduate School of Environment and Information Sciences, Yokohama National University, Yokohama-shi, 240-8501 Japan.

[††] The author is with Cybersecurity Technology Institute, Institute for Information Industry, 105 Taipei, Taiwan.

[†††] The authors are with Institute of Advanced Sciences, Yokohama National University, Yokohama-shi, 240-8501 Japan.

a) E-mail: wu-chun-jung-zt@ynu.jp

DOI: 10.1587/transcom.2019CPP0009

similar infection commands. Moreover, through similarity analysis of command sequences, we can identify the malware category of unknown threats.
2. By clustering telnet logs, we discovered a new DoS cyberattack executed using pure Linux commands, without IoT malware binaries.
3. Using malware samples from the IoT honeypot, our proposed method could identify malware categories with 96.70% accuracy.

2. Related Works

In this section, we review the literature on the malware classification problem. Yen et al. (2013) conducted an epidemiological study of malware encountered in a large, multinational enterprise. They collected security and network infrastructure logs to determine the key behavioral features of web-based malware. Moreover, they used a logistical regression model to identify and rank the malware risk [5]. Masud, Khan, and Thuraisingham presented a method of detecting malicious executables that combined three types of features: binary N-grams, assembly instruction sequences, and dynamic-link library function calls [6].

In 2015, Microsoft and Kaggle held the Malware Classification Challenge (BIG 2015), in which Microsoft provided 20,000 Windows malware binary and assembler code files, with nine categories of malware. Contestants had to classify the malware categories as well as possible. The winning team extracted different features from the ASM file opcode and gathered pixel data from malware disk images, then applied an N-gram algorithm to predict the malware category, thereby achieving 99.7% accuracy. Ahmadi et al. subsequently used similar features to improve the classification algorithm and achieve 99.8% accuracy with lower computational costs [7].

Drew, Moore, and Hahsler applies the Strand gene sequence classifier, which offers a robust classification strategy that easily accommodates unstructured data, to malware classification. Their method was used on approximately 500 GB of data to predict nine polymorphic malware categories, and the results indicated that, with minimal adaptations, it achieved an accuracy of well over 95% [8]. Most research has analyzed Windows-based malware and devised experiments in MS Windows platforms.

For Linux/Unix malware, Shahzad and Farooq analyzed 709 Linux executable and linkable format (ELF) files, extracting features from the ELF header and then applying machine-learning classifiers to detect malware. Their method achieved 99% detection accuracy, with a false alarm rate of less than 0.1% [9]. Bai et al. gathered features from ELF file system calls and tested four classification algorithms (J48, Random Forests, AdboostM1, and IBK) for detecting Linux malware, achieving a detection accuracy of approximately 98% [10].

Given that serious worm attacks have occurred through the Internet, Wang et al. proposed a worm detection method based on mining dynamic program executions. They analyzed system calls from MS Windows and Linux and traced system call sequences using a natural language processing algorithm. They also applied the machine-learning algorithms Naive Bayes and Support Vector Machines (SVM), with SVM achieving a 99.5% worm detection rate and a 2.22% false positive rate [11].

For Android malware, Ham et al. [12] extracted features about the network, phone, message, CPU, battery, and memory for each process in Android devices. They apply a linear SVM to detect Android malware and compare the malware detection performance of SVM with that of other machine learning classifiers. They show that the SVM outperforms other machine learning classifiers with 0.995 Accuracy and 0.957 Precision.

Azmoodeh, Dehghantanha, and Choo [13] presented a deep learning based method to detect Internet of Battlefield Things (IoBT) malware via the device's Operational Code (OpCode) sequence. They transmuted OpCodes into a vector space and apply a deep Eigenspace learning approach to classify malicious and benign application. Their method could achieve 99.68% accuracy and 98.37% recall.

Su et al. [14] proposed a novel lightweight method of detecting DDoS malware in IoT environments. First, one-channel grayscale images converted from binaries were extracted, and then a lightweight convolutional neural network was used to classify IoT malware families. The experimental results indicated that this system could achieve 94.0% accuracy in goodware and DDoS malware classification and 81.8% accuracy in classification of goodware and the two main malware families.

Our study examined Linux malware. Its major difference from other research lay in the dataset. We primarily analyzed shell commands from IoT malware, also examining the file meta-information when necessary.

3. Methods

3.1 Preliminaries

All the data in this research were observed in IoTPOT [3]. We used VirusTotal malware labels as the ground truth of data. Command sequences were extracted from IoTPOT telnet session logs and concatenated into a sequence according to the input order. A command sequence could contain single or multiple shell command clauses. In this study, we mainly analyzed infection sequences according to target, purpose, and frequency, and found that most of them consisted of five types of atomic behavior:

1. Authentication behavior
Login with ID and password
2. Resource scan behavior
Resource scan is a command that finds available functions and writable folders in an IoT device; for example, "/bin/busybox Mirai" tests "/bin/busybox," and "/bin/busybox cat /proc/mounts" aims to find a writable

folder.

3. Change directory behavior
   Changes the directory/folder of the terminal's shell.
4. Create or download files behavior
   Uses "echo" to produce binary files or "wget," "tftp" to download files.
5. Execution behavior
   Uses "sh" to execute downloaded binary or script files.

Malware sometimes executes "chmod" to alter file privileges, "history –c –r" to purge the system log, "rm" to remove files, and "exit" to terminate the session. These commands may be executed multiple times to ensure that the infection is successful. Only a few "kill" and "killall" commands were found in our data, they mostly tighter with the "while true;" loop and "wget" commands.

We collected 69 million command sequences containing the Mirai signature, all of the sequences were recognition type, which only contains login credentials and several Linux commands such as "enable, system, shell, sh, and /bin/busybox echo." These commands were not related to malware binaries, and the information of each sequence was too few for analysis. IoTPOT did not capture any Mirai infection command sequences because dozens of verification steps are performed by the attacker's server, each of which must receive a corresponding response, such as the "echo," "cat /etc/mounts," and "cp /bin/echo" commands. We determined these steps based on the Mirai source code [15], [16]. To observe Mirai's infection command sequence, we developed new honeypots consisting of real IoT devices that could respond correctly to Mirai. We thereby captured 578,671 Mirai command sequences from December 11, 2016, to February 28, 2017, approximately 32% of which were infection command sequences executed by Mirai and 68% of which were simple recognition command sequences. We, therefore, had to narrow the scope of the command sequences, focusing on those that related to downloaded malware binaries or cause the serious cyber security threat. Table 1 shows the results of other research [4] with a comparison of labels.

In this study, VirusTotal was used to obtain scan results from 66 antivirus engines. We sent 12,821 unique malware MD5s from IoTPOT and received 3,306 reports. We then chose the most frequent malware family name as the representative malware category. Table 2 shows the top five antivirus engines for IoT malware.

We chose Kaspersky, DrWeb, and ESET-NOD32 to locally scan 40,203 different IoT malware binaries and found that DrWeb could label 39,245 of them, representing 97.61% of the submitted malware and thus surpassing Kaspersky (69.82%) and ESET-NOD32 (74.57%). Therefore, we employed DrWeb to label the IoT malware as the basis for malware categories.

Table 1 Comparison of labels and infection command sequence [4].

Label	Pattern of Command Sequence	Number of attack/Day (Average)
Bashlite	Using a downloaded shell script, kill previously running malicious process, download malware binaries of different CPU architectures, and block 23/TCP to prevent other infections. Run all downloaded malware binaries.	290
Nttpd	Change directory to /tmp. Check whether shell can be used by echoing "welcome." Run binaries.	48
ZORRO	Check whether "wget" command is usable. Check whether busybox shell can be used by echoing "ZORRO." Remove various commands and files under /usr/bin/, /bin, /var/run/, and /dev. Copy /bin/sh to random filename. Append series of binaries to random filename and make attacker's own shell. Using attacker's own shell, download binary. The IP address and port number of the malware download server can be seen in the command. Run binaries.	2232

Table 2 Top 5 antivirus engines for IoT malware.

AV/Consistency %	Mirai (207)	Bashlite (2733)	Hajime (5)	Tsunami (48)
Kaspersky	98.06	100	100	89.58
DrWeb	96.61	97.36	100	60.41
ESET-NOD32	98.55	93.66	100	91.66
Sophos	85.50	89.82	80	95.83
Avast	85.02	88.58	20	35.41

Table 3 An example of the command mapping table.

Command	token	Command	token
/bin/busybox	B	exit	q
cd	C	chmod	c
enable	e	echo	E
sh	h	wget	w

3.2 Encoding and Measurement of Command Sequences

Data encoding. To process numerous complex sequences we used a simplified representative form called extracted command tokens (ECTs). For example, sequences of the command ['cd /tmp || cd /var/run || cd /dev/shm || cd /mnt || cd /var; tftp -r tftp.sh -g test.test.org; sh tftp.sh; busybox wget http'] can be expressed as the encoded sequence "cccccctsw," which represents each command by a single letter, such as "w" for "wget" and "c" for "cd." Then applying a natural language processing algorithm to classify the ECTs, having made a table mapping each of the 51 commands to a corresponding letter. An example of a command mapping table is shown in Table 3. These commands were derived from historical observation data in the IoTPOT.

Comparison of distance measures. The following six distances [17] are applied to measure similarity between different categories.

1. Cosine: Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them.

2. Trigram: Apply trigram distance to determine how similar two strings are.
3. Normalized longest common subsequence (NLCS): The longest common subsequence can be considered the sequential analog of the cosine distance between two ordered sets.
4. Metric longest common subsequence (MLCS): Measure the degree of similarity between the two series.
5. Normalized Levenshtein Distance (NLD): The sum of the length normalized Levenshtein distance between the words occupying the same meaning slot divided by the number of word pairs.
6. Jaro-Winkler: A string metric measuring an edit distance between two sequence with favorable ratings to strings that match from the beginning for a set prefix length.

We choose the top 500 command sequences in the four categories and then calculated the average and minimum distances between categories, with the results shown in Table 4 and Table 5. Specifically, we apply six distance measurement methods and calculate the adjacency matrix between pairs of malware categories, and then calculate the average and minimum distance. The columns under the "B-N" header indicate the distance between Bashlite and nttpd using the six measurements, the columns under "M-Z" the distance between Mirai and ZORRO, and so on. We determined that cosine was the best distance for distinguishing ECTs, with trigram just slightly lower. However, we found a few command sequences in the same session containing both Bashlite and nttpd. The command sequence from the Bashlite source code of has been leaked, so that other attackers can copy its function and use it in their malware [18]. Therefore, the malware executes the command sequence using multiple categories' signatures. Because cosine distance cannot distinguish this type of Bashlite command sequence, trigram provides a better solution for combined or mixed command sequences. Moreover, the source code for Mirai has similarly been leaked [19]. Based on the distance measure results, we chose trigram for use in this study.

Table 4 Distance measures for different malware labels (average).

Distance/category	B-N	B-M	B-Z	N-M	N-Z	M-Z
Cosine	0.9686	0.9492	0.9958	0.9990	1.0000	0.9516
Trigram	0.9086	0.9436	0.9230	0.9516	0.9661	0.9351
NLCS	0.7864	0.8888	0.8621	0.8895	0.9266	0.8642
MLCS	0.8367	0.9307	0.8971	0.9299	0.9472	0.9132
NLD	0.8795	0.9358	0.9118	0.9336	0.9585	0.9197
Jaro-Winkler	0.5238	0.6001	0.5615	0.5457	0.5851	0.4710

Table 5 Distance measures for different malware labels (minimum).

Distance/category	B-N	B-M	B-Z	N-M	N-Z	M-Z
Cosine	0.0163	0.7259	0.2929	0.5736	0.9678	0.1783
Trigram	0.4048	0.6968	0.1111	0.6667	0.6481	0.6197
NLCS	0.2500	0.5748	0.1429	0.4118	0.5556	0.3734
MLCS	0.3529	0.5893	0.1667	0.5455	0.6000	0.4048
NLD	0.3929	0.6857	0.1667	0.5455	0.6667	0.5318
Jaro-Winkler	0.2167	0.3217	0.0978	0.3119	0.2933	0.2251

Malware category Feature extraction. N-gram is an algorithm based on computational linguistics and probability [20], which can be used to estimate the likelihood of a sentence occurring at all or following a given word. N-gram can also provide efficient approximate string matching. Using N-gram to index lexicon terms, a signature file can be compressed to a smaller size. Moreover, N-gram can be used to calculate the similarity between two strings [21].

In this study, we used N-gram to collect ECT occurrence patterns. For each malware category, we collected the top 10 N-gram features, representing the major behavior in each category, and presented them as a histogram. These features were all based on a trigram model, namely, n = 3. We calculated the trigram histogram using four months data for two categories. The command sequence distribution of Bashlite, Mirai, and Satori are shown in Figs. 1, 2, and 3, respectively. Based on the common command patterns of each malware category, we found that Bashlite tended to contain "cd," "sh," and "rm" behavior, Mirai often contained terms such as "busybox," "cat," and "echo," and Satori tended to use terms such as "&&," "cp," and "busybox." The results indicated that trigram could assist in revealing distinctive attack patterns among different IoT malware categories.

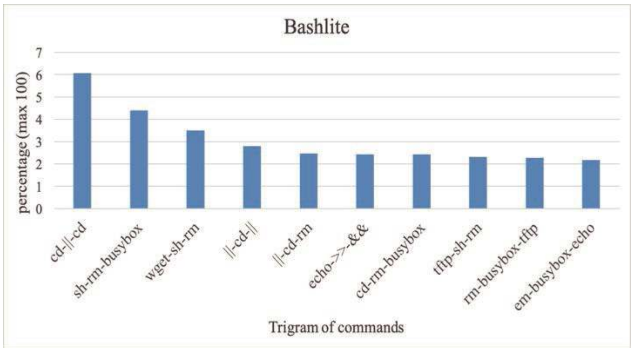


Fig. 1 Trigram statistics of Bashlite IoT malware.

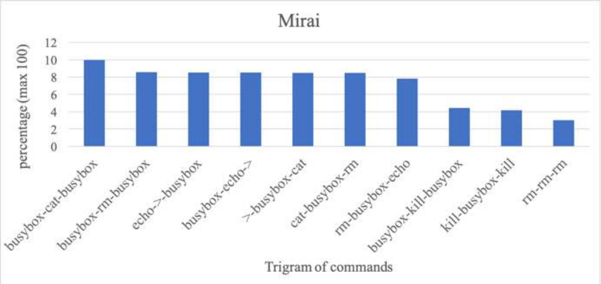


Fig. 2 Trigram statistics of Mirai IoT malware.

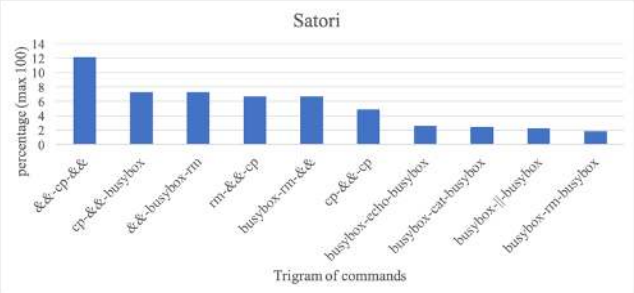


Fig. 3 Trigram statistics of Satori IoT malware.

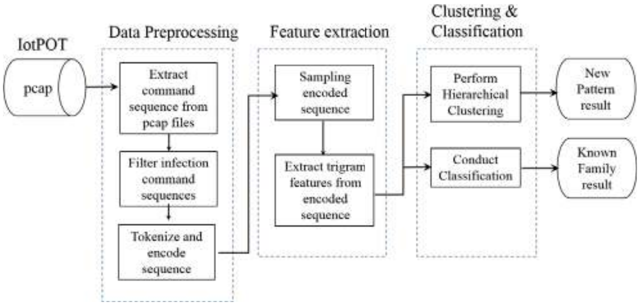

IotPOT
pcap
Data Preprocessing
Extract command sequence from pcap files
Filter infection command sequences
Tokenize and encode sequence
Feature extraction
Sampling encoded sequence
Extract trigram features from encoded sequence
Clustering & Classification
Perform Hierarchical Clustering
Conduct Classification
New Pattern result
Known Family result

Fig. 4 Data analysis flow.

3.3 Data Analysis

The complete analytical process is illustrated in Fig. 4. First, command sequences were extracted from pcap files, filtering for infection command sequences. Next, the command sequence was encoded to create ECTs, and then the N-gram model was used to extract trigram features from them. Finally, classification and clustering analysis was performed to determine malware categories and new patterns.

Clustering algorithms. The IoT cyberattacks are keep evolving and the pattern of attacks is uncertain. Therefore, we choose the hierarchical clustering method because it does not require to predefine the number of clusters. To identify new patterns, we chose a single-linkage hierarchical clustering algorithm sensitive to outliers. The hierarchical clustering method works by successively combining individual data into cluster [22]. To our knowledge, the use of clustering algorithms in malware-related datasets was introduced Bailey et al. [23], who also employed hierarchical clustering.

Classification algorithms. We have transformed the telnet logs into a smaller ECTs' dataset. However, trigram still generated hundreds of features. According to our statistics of trigrams, some Linux commands occurrence tightly in order, such as malware would like to utilize "cd" to move to a writable folder and then conduct the download tasks via "wget" or "tftp" commands. In text categorization research, Joachims [24] has shown SVM could handle high dimensional input space and few irrelevant features. Instead, The Naive Bayes classifier assumes that the distribution of different terms is independent of each other. Even though the independence assumption is false in many real-world applications, Naive Bayes performs surprisingly well [25]. Therefore, we chose SVM as our classification algorithm and Naive Bayes classifier as the baseline. After conducting the same experiments with these two algorithms, we can determine which was better for our research.

- Naive Bayes

Naive Bayes classifiers assume that an attribute value's effect on a given class is independent of the values of other attributes; this is called "class-conditional independence" [26]. The Naive Bayes classifier greatly simplifies learning by assuming that features are independent given class. Although independence is generally a poor assumption, in practice Naive Bayes often competes well with more sophisticated classifiers [27]

- Support Vector Machines

SVM is a useful technique of data classification, whose aim is to produce a model that predicts the target values of the test data based solely on their attributes [24]. Given a training set of instance–label pairs (x_i, y_i), $i = 1, \ldots, l$, where $x_i \in R^n$ and $y_i \in \{1, -1\}$, SVM requires solution of the following optimization problem:

$$\min_{w,b\zeta} \frac{1}{2} w^T w + c \sum_{i=0}^{l} \xi_i \tag{1}$$

$$\text{Subject to } y_i \left(w^T \varphi(x_i) + b\right) \geq 1 - \xi_i,\ \xi_i \geq 0,\ i = 1, \ldots, l$$

In Eq. (1), $K\ (x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$ is called the kernel function. Many kinds of kernel function options are available, such as linear, polynomial, and sigmoid. For our dataset, we chose a linear kernel function according to our ECT numbers [23], [24].

Classification evaluation. We used a confusion matrix and accuracy to measure the classification result. Given a target category, let TP (true positive) be the number of ECTs correctly classified as the target category; FN (false negative) the number of sequences from the target category misclassified as another; TN (true negative) the number of sequences from other categories that are correctly classified; and FP (false positive) the number of sequences incorrectly classified as the target category. The precision (P) is defined as precision = TP/ (TP + FP), and the recall rate (R) is defined as recall = TP/ (TP + FN). The F-score represents the harmonic mean of (P) and (R) and provides a balance between them: F-score = 2 PR/ (P + R). The F-score assists in identifying a threshold of similarity. Accuracy (A) is defined as accuracy = (TP+TN) / (TP + FP + FN + TN), and the error rate is defined as error rate = (FP+FN) / (TP + FP + FN + TN) [22].

4. Experiments

4.1 Dataset and Environment

Data collection for classification was undertaken from December 2016 to September 2017. The dataset contained data

Table 6 Dataset for analysis.

Dataset	Number of infection command sequence	Rearranged command sequence	Number of unique ECTs	Time interval	Analysis
1	2,756,231	44,843*	2,925	2016/12/07~ 2017/09/16	NB, SVM
2	422,591	95,448**	4,626	2017/04/01~ 2017/04/30	Hierarchical clustering

* Correlating malware binaries, extracting command sequences that download binaries, and conducting data deduplication
** Extracting command sequences and conducting data deduplication

Table 7 Malware categories and ECTs' distribution.

Label	Bashlite	Mirai	Hajime	Tsunami
Numbers of ECTs	665	3408	155	58
Numbers of malware	2755	2665	34	162

Table 8 Statistics of time cost.

Algorithm	Set	Data preprocessing	Feature extraction	Clustering/ Classification	Total
SVM	1	174 mins 1 secs	21 secs	20 secs	174 mins 42secs
Naïve Bayes	1	174 mins 1 secs	21 secs	15 secs	174 mins 37 secs
Hierarchical Clustering	2	13 secs	73 mins 21 secs	1 mins 5 secs	74 mins 39 secs

for 284 days from real IoT devices in the IoTPOT. As illustrated in Table 6, the dataset included 2.7 million infection command sequences related to malware that we downloaded to another server in real time. These sequences could be reduced to 44,843 unique command sequences through correlation and deduplication. Moreover, our encoding method was able to reduce the command sequences to 2,925 ECTs. To discover hidden patterns, we chose the data for a one-month period as dataset 2 for the clustering experiment.

DrWeb was used to scan the malware binaries, after which malware labels and corresponding ECTs could be obtained. The distribution of labels is shown in Table 7, indicating that the majority of malware in the IoTPOT came from Bashlite and Mirai.

From Dec. 7th, 2016 to Sept. 16th, 2017, our honeypot has collected 1.36 terabyte (TB) pcap files. We extract 22.9 gigabytes (GB) telnet logs via a server with ten cores Intel 2.20 GHz CPU, 62 GB RAM, and 4 Terabytes disk. This task is scheduled automatically run every day. Processing 1.36 TB pcap files will cost about eight days. The other time cost of our method is shown in Table 8. The data preprocessing begins at filter infection command from telnet logs. For filter out and label the malware related telnet logs in dataset 1, we utilize Google BigQuery [28] to process 22.9 GB telnet logs. ECTs transformation and machine learning algorithms were conducted via a machine with two quad-core Intel 3.70 GHz CPU, 16 GB RAM, and 1 TB disk. The SciPy 0.18.1 [29] is used for supporting clustering and classification algorithms.

4.2 Clustering Experiments

The hierarchical clustering method involves successively combining individual data into clusters. We conducted a hierarchical clustering analysis using dataset 2. As shown in Fig. 5, the algorithm separated 4,636 ECTs into 30 clusters according to the trigram distance. We labeled the clusters according to antivirus engine scan results of malware binaries or with reference to malware analysis reports from cybersecurity researchers. Detail text features of malware families are summarized in Appendix A.

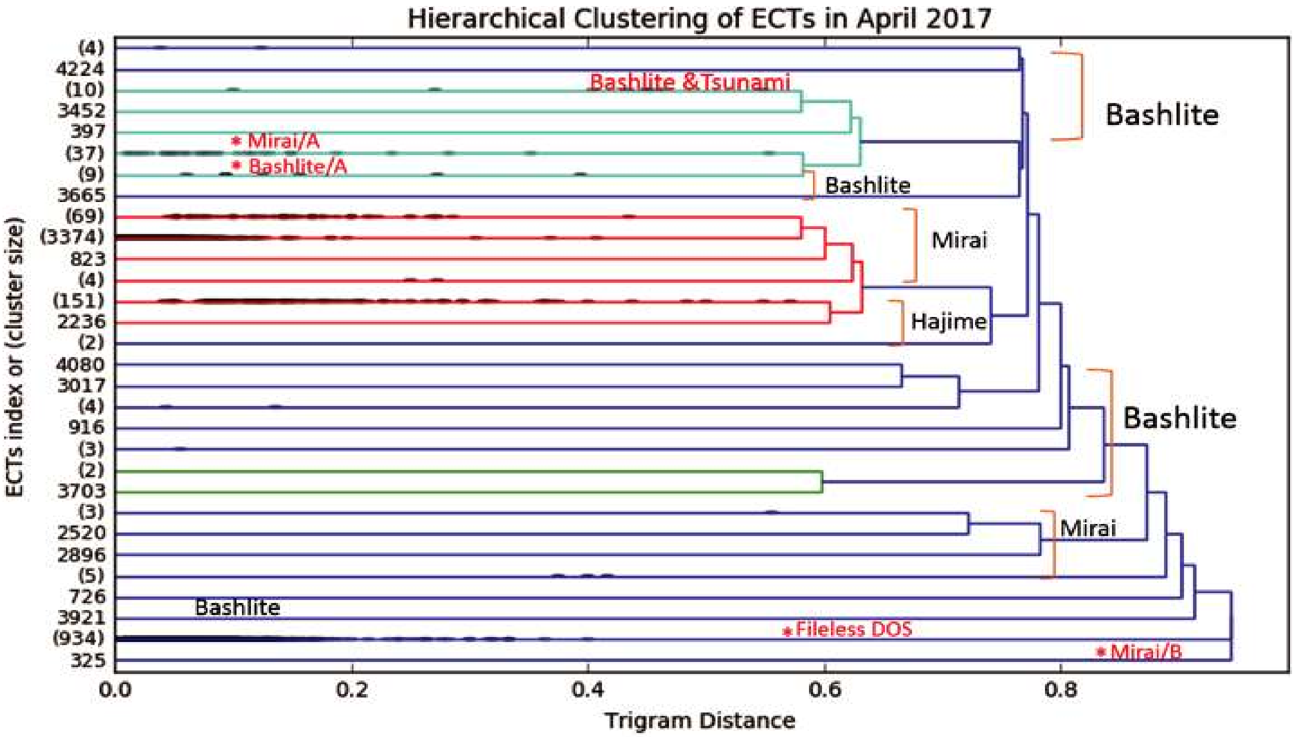
Our method successfully differentiated three known malware families and their variants and also discovered one new IoT cyberattack pattern, called "Fileless DoS." Although the four best-known malware families are Mirai, Bashlite, Hajime, and Tsunami, Tsunami employs Linux commands in a similar manner to Bashlite and be assigned to the leaf cluster named "(10)". The MD5 of Tsunami which shares similar infection pattern of Bashlite, shown in Appendix B.

The clustering results helped to discover the following malware variants and new cyberattack pattern:

- Mirai/A and Bashlite/A are malware variants that truncate ptmx files after login. The ptmx file is used to create a pseudoterminal master–slave pair [30]. Both Mirai/A and Bashlite/A contain this command, and the maximum distance to separate them must be less than 0.58.
- Mirai/B targets devices with weak default credentials which login ID is root or Admin and password is 5up, such as TP-Link (TL-WR740N) [31]. The Mirai/B commands are more straightforward than those of the original Mirai. For example, Mirai/B does not check device partitions such as cat /proc/mounts [16].
- Fileless DoS is a shell script that employs an infinite while loop and multiple wget commands to mount a DoS attack. Downloaded web contents are sent to /dev/null, and thus no binaries are stored in devices. A total of 934 Fileless DoS ECTs were discovered in April 2017. The top ten victim websites are shown in Table 9, including those of a music band, a construction company, and an IT solutions company.

4.3 Classification Experiments

Because the data amounts varied greatly among categories, we designed two experiments to identify whether data bias affected classification accuracy. Two datasets were prepared for our experiments. The first contained 1000 ECT types per malware category; Bashlite, Hajime, and Tsunami data were repeated up to 1000. The second dataset contained all ECT types from every category. Our program randomly chose 50% of the data as a training set and then tested the remaining 50%. To avoid selecting only Mirai data, however, we



X-axis: ECT index number # or cluster size (*)

Fig. 5 Labeled hierarchical clustering results of ECTs in April 2017.

Table 9 Victims of Fileless DoS.

Victim websites	Counts
http://fxxxxxxxx.com:80	7111
http://xxx.xxx.80.118:80	5669
http://www.txxxxxxxxxxxx.com:80	2722
http://www.hxxxxxxxx.co.il:80	2564
http://www.bxxxxxxxxxxxxxxxxxxxxxx.com:80	2354
http://www.kxxxxxxxxxxxxxxxxxxxxxxxxxx.de:80	1982
http://txxxxxxxxxxx.com:80	1980
http://www.axxxxx.dk:80	1878
http://xxx.xxx.19.69:80	1843
http://cxxxxxxxxxxxxxxxxxxx.com:80	1749

Table 10 Classification performance of even sampling- Naive Bayes.

	precision	recall	f1-score	support
Bashlite	0.87	0.89	0.88	513
Mirai	0.83	0.77	0.80	499
Hajime	1.00	1.0	1.00	476
Tsunami	0.70	0.74	0.72	512
avg / total	0.85	0.85	0.85	2000

Table 11 Classification performance of even sampling- SVM.

	precision	recall	f1-score	support
Bashlite	1.0	0.87	0.93	513
Mirai	0.85	0.75	0.80	499
Hajime	1.0	1.0	1.0	476
Tsunami	0.70	0.87	0.78	512
avg / total	0.89	0.87	0.87	2000

Table 12 Classification performance of random sampling- Naive Bayes.

	precision	recall	f1-score	support
Bashlite	0.88	0.93	0.91	155
Mirai	0.98	0.99	0.99	1225
Hajime	0.94	1.0	0.97	60
Tsunami	0.00	0.00	0.00	24
avg / total	0.95	0.97	0.96	1464

Table 13 Classification performance of random sampling- SVM.

	precision	recall	f1-score	support
Bashlite	0.92	0.94	0.95	155
Mirai	0.98	1.0	0.99	1225
Hajime	1.0	1.0	1.0	60
Tsunami	0.00	0.00	0.00	24
avg / total	0.96	0.98	0.97	1464

randomly chose the training dataset for the second experiment. The precision and recall scores are listed as Table 10, Table 11, Table 12, and Table 13.

Based on the results of these experiments, SVM performed better than Naive Bayes. However, Tsunami was easily misclassified as Bashlite. We believe that second-stage training is necessary for real cases. Such reinforcement learning — also called active learning — involves fine-tuning the model during the training process. Therefore, based on the prediction results for Bashlite and Tsunami, we further developed a sub-training approach by adding an additional feature (file size) and performing sub-classifier

Table 14 Precision/recall of SVM – second stage (reinforcement learning).

	precision	recall	f1-score	support
Bashlite	0.99	0.99	0.99	155
Mirai	0.98	1.00	0.99	1225
Hajime	1.0	1.0	1.00	60
Tsunami	0.90	0.86	0.88	24
Avg / total	0.96	0.98	0.97	1464

training. As shown in Table 14, the precision of Tsunami classification improved because its file sample metadata differed from that of Bashlite. Using additional features can thus help to prevent misidentifying classes that share the same command line pattern, without requiring static and dynamic analyses and simply by looking at the command line and file meta-information. Mirai's open source code provides hackers with an entry point for developing new variants. It has been noted that hackers rely on using known or zero-day vulnerabilities for developing new Mirai variants to attack IoT devices [32]. Hence, these evaluations may incur new patterns of ECTs.

5. Discussion

For IoT malware which attacks via the telnet protocol, our clustering experiments show our method can find new cyberattack, "Fileless DoS" and changes from malware variants. Moreover, our trigram features could help classification of IoT malware. Comparisons with previous studies are as follows:

- The method proposed by Ham et al. [12] rely on features about the network, phone, message, CPU, battery, and memory for each process in Android devices. However, IoT devices are hard to extend and install third-party packages. Our method only analyzes the telnet traffic between attacker and victim devices. There is not any modification for IoT devices.
- Azmoodeh, Dehghantanha, and Choo [13] analyzed the OpCode sequence and applied a deep Eigenspace learning approach to classify malicious and benign application. Their method is excellent that could achieve 99.68% accuracy. The OpCode sequence generated by malware binaries, but IoT malware, such as Mirai and Bashlite may remove their binaries after execution. Moreover, many IoT devices utilized flash storage, rebooting will erase the malware binaries. However, our method does not need to convert binaries to OpCode and graph, can infer the malware family by telnet traffic and demonstrates 96.70% accuracy.
- Su et al. [14] investigated a lightweight method of detecting DDoS malware in IoT environments. They converted binaries to grayscale images and then classified IoT malware families by a convolutional neural network. The system could achieve 94.0% accuracy in goodware and DDoS malware classification and 81.8% accuracy in classification of goodware and the two main malware families. Su's method only examines Mirai and Bashlite family. Our method examines four malware families and achieves 96.70% accuracy.

In this paper, we utilize real IoT devices as honeypot to obtain the dataset. These devices are known to have been targeted by IoT malware and in that sense we believe that the dataset can provide partial view of real cyberattacks against IoT devices in the wild. We cannot claim that the dataset represents the whole attacks in IoT as we have only limited number of devices for honeypot. However, we believe the study is meaningful as the honeypot was indeed able to observe and capture samples from four major IoT malware families targeting IoT telnet services and the proposed method was able to discover evolving attack like fileless DoS.

The limitations of this paper may come from the attack vector of IoT malware: (1) our method does not analyze HTTP or SSH protocol, and (2) our method might be affected if hackers intentionally add parts of other malware codes to their malware.

6. Conclusion

The confusion tables and the accuracy of our classification method led to several clear conclusions. First, the lowest accuracy of all the ECTs was 0.9675, indicating that even for a dataset spanning eight months our method remained valid. Although command sequences can change many times, the use of trigram features can properly distinguish Mirai, Bashlite, and Hajime malware, based on differences in their infection command patterns. These malware categories have distinctive command patterns and the hidden feature can be extracted for further analysis. Second, we demonstrated that using clustering with a trigram sequence can detect variant attack patterns (for example, wget DoS attack) and facilitate identification of similarities between different malware families, without requiring the collection of malware binaries.

Although the present study has yielded findings with both theoretical and practical implications, its design is not without shortcomings. However, despite this study's limitations, we conclude that applying a text-mining algorithm to malware behavior analysis is effective. Future research would include (1) identifying the target device type, (2) identifying malware variant and feedbacking to malware family classification, and (3) analyzing the common network traffic behavior of IoT malware.

7. Future Work

Many different adaptations, tests, and experiments have been left for the future due to lack of time. Future work concerns more in-depth analysis of the optimal clustering algorithm and efficient ways to put the proposed method in practices as follows.

1. To find the best clustering algorithm, we should implement and conduct experiments to compare the quality of different clustering algorithms.

2. To verify if the trained classifier could act as an instruction detection system.

The step of extracting data from pcap files is time costly. However, if the proposed method can deal with TCP flow directly, which can economize time. As an example, IT infrastructure engineers could mirror telnet flow or deploy our classifier at the Man-in-the-middle proxy. These telnet logs and network traffic header information were used to compare the similarity with our trained model. Therefore, we can examine if the trained classifier could act as an instruction detection system which indicate suspicious connections using malicious command sequences and also the corresponding malicious categories.

Acknowledgments

A part of this work was obtained from EU-Japan collaboration project “Multi-layered Security technologies to ensure hyperconnected smart cities with Blockchain, Big Data, Cloud and IoT (MSEC).”

References

[1] P. Loshin, “Details emerging on Dyn DNS DDoS attack, Mirai IoT botnet,” TechTarget network, http://searchsecurity.techtarget.com/news/450401962/Details-emerging-on-Dyn-DNS-DDoS-attack-Mirai-IoT-botnet, accessed Jan. 18. 2019.

[2] M. Kuzin, Y. Shmelev, and V. Kuskov, “New trends in the world of IoT threats,” AO Kaspersky Lab, https://securelist.com/new-trends-in-the-world-of-iot-threats/87991/, accessed Feb. 6. 2019.

[3] L.J. Rivera and L. Goasduff, “Gartner says a thirty-fold increase in internet-connected physical devices by 2020 will significantly alter how the supply chain operates,” Gartner, https://www.gartner.com/newsroom/id/2688717, accessed Jan. 18. 2019.

[4] Y.M.P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, “IoTPOT: A novel honeypot for revealing current IoT threats,” J. Information Processing, vol.24, no.3, pp.522–533, 2016.

[5] T.F. Yen, V. Heorhiadi, A. Oprea, M.K. Reiter, and A. Juels, “An epidemiological study of malware encounters in a large enterprise,” Proc. 2014 ACM SIGSAC Conference on Computer and Communications Security, ACM, pp.1117–1130, 2014.

[6] M.M. Masud, L. Khan, and B. Thuraisingham, “A scalable multi-level feature extraction technique to detect malicious executables,” Inform. Syst. Front., vol.10, no.1, pp.33–45, 2008.

[7] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto, “Novel feature extraction, selection and fusion for effective malware category classification,” Proc. Sixth ACM Conference on Data and Application Security and Privacy, ACM, pp.183–194, 2016.

[8] J. Drew, T. Moore, and M. Hahsler, “Polymorphic malware detection using sequence classification methods,” Proc. 2016 IEEE Security and Privacy Workshops (SPW), 2016.

[9] F. Shahzad and M. Farooq, “ELF-Miner: Using structural knowledge and data mining methods to detect new (Linux) malicious executables,” Knowledge and Information Systems, vol.30, no.3, pp.589–612, 2012.

[10] J. Bai, Y. Yang, S. Mu, and Y. Ma, “Malware detection through mining symbol table of Linux executables,” Information Technology J., vol.12, no.2, pp.380–384, 2013.

[11] X. Wang, W. Yu, A. Champion, X. Fu, and D. Xuan, “Detecting worms via mining dynamic program execution,” Proc. 2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops - SecureComm 2007, pp.412–421, 2007.

[12] H.-S. Ham, H.-H. Kim, M.-S. Kim, and M.-J. Choi, “Linear SVM-based android malware detection for reliable IoT services,” J. Applied Mathematics, vol.2014, 2014.

[13] A. Azmoodeh, A. Dehghantanha, and K.-K.R. Choo, “Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning,” IEEE Trans. Sustain. Comput., vol.4, no.1, pp.88–95, 2019.

[14] J. Su, D.V. Vargas, S. Prasad, D. Sgandurra, Y. Feng, and K. Sakurai, “Lightweight classification of IoT malware based on image recognition,” Proc. 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, Japan, 2018.

[15] B. Krebs, “Source code for IoT botnet ‘Mirai’ released,” https://krebsonsecurity.com/2016/10/source-code-for-iot-botnet-Mirai-released/, accessed Jan. 18. 2019.

[16] J. Gamblin, “Mirai-Source-Code,” GitHub, https://github.com/jgamblin/Mirai-Source-Code/, accessed Jan. 18. 2019.

[17] W.H. Gomaa and A.A. Fahmy, “A survey of text similarity approaches,” International Journal of Computer Applications, vol.68, no.13, pp.13–18, 2013.

[18] A. Tellez, “Bashlite,” GitHub, https://github.com/anthonygtellez/BASHLITE, accessed Jan. 18. 2019.

[19] D. Davidson, “linux.mirai,” https://github.com/0x27/, accessed Jan. 18. 2019.

[20] G. Kondrak, “N-gram similarity and distance,” International Symposium on String Processing and Information Retrieval, pp.115–126, Springer Berlin Heidelberg, 2005.

[21] P.F. Brown, P.V. Desouza, R.L. Mercer, V.J.D. Pietra, and J.C. Lai, “Class-based n-gram models of natural language,” Computational Linguistics, vol.18, no.4, pp.467–479, 1992.

[22] I.H. Witten, E. Frank, and M.A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, 2016.

[23] M. Bailey, J. Oberheide, J. Andersen, Z.M. Mao, F. Jahanian, and J. Nazario, “Automated classification and analysis of internet malware,” Recent Advances in Intrusion Detection, C. Kruegel, L. Lippmann, and C Andrew, eds., Lecture Notes in Computer Science, vol.4637, pp.178–197, 2007.

[24] T. Joachims, “Text categorization with support vector machines: Learning with many relevant features,” European Conference on Machine Learning, pp.137–142, 1998.

[25] A. Hotho, A. Nürnberger, and G. Paaß, “A brief survey of text mining,” LDV Forum, vol.20, no.1, pp.19–62, 2005.

[26] K.P. Murphy, Naive Bayes Classifiers, University of British Columbia, Ph.D. Thesis, 2006.

[27] I. Rish, “An empirical study of the naive Bayes classifier,” IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, vol.3, no.22, pp.41–46, 2001.

[28] K. Sato, “An inside look at google bigquery,” White paper, Google, https://cloud.google.com/files/BigQueryTechnicalWP.pdf, accessed Jan. 18. 2019.

[29] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” J. Machine Learning Research, vol.12, pp.2825–2830, Oct. 2011.

[30] die.net, “ptmx(4) - Linux man page,” https://linux.die.net/man/4/ptmx, accessed Jan. 18. 2019.

[31] J. Trost, “7up (Mirai?) Triage, More IoT Malware Targeting Weak Passwords,” http://www.covert.io/7up-mirai-triage-more-iot-malware-targeting-weak-passwords/, accessed Jan. 18. 2019.

[32] C. Zheng, C. Xiao, and Y. Jia, “IoT malware evolves to harvest bots by exploiting a zero-day home router vulnerability,” Palo Alto Networks, https://researchcenter.paloaltonetworks.com/2018/01/unit42-iot-malware-evolves-harvest-bots-exploiting-zero-day-home-router-vulnerability/, accessed Jan. 18. 2019.

Appendix:

Table A· 1 The text features and family of clusters.

<table>
<tr><th>Feature Id</th><th>ECT index/ cluster size (*)</th><th>Cluster id</th><th>Text features</th><th>Family</th><th>Numbers of ECTs</th></tr>
<tr><td>1</td><td>(4)</td><td>25</td><td>rm -rf ; pkill -9 ;killall -9 ;, shell, cd /tmp || cd /var/system || cd /mnt || cd /lib;rm -f /tmp/ || /var/run/ || /var/system/ || /mnt/ || /lib/*</td><td>Bashlite</td><td>4</td></tr>
<tr><td>2</td><td>4224</td><td>26</td><td>cd /tmp || cd /var || cd /dev/shm || cd /var/tmp || cd /root || cd /, wget/tftp/get/ftpget, chmod, sh, rm</td><td>Bashlite</td><td>1</td></tr>
<tr><td>3</td><td>(10)</td><td>21</td><td>cd /tmp || cd /var || cd /dev/shm || cd /var/tmp || cd /root || cd /, wget/tftp/get/ftpget, chmod, sh, rm</td><td>Bashlite (&amp; Tsunami)</td><td>10</td></tr>
<tr><td>4</td><td>3452</td><td>22</td><td>>/dev/netslink/.t && cd /dev/netslink/</td><td>Bashlite</td><td>1</td></tr>
<tr><td>5</td><td>397</td><td>23</td><td>cd /tmp; rm -fr *; wget/curl/tftp/get/ftpget, chmod, sh, rm</td><td>Bashlite</td><td>1</td></tr>
<tr><td>6</td><td>(37)</td><td>20</td><td>'>/tmp/.ptmx && cd /tmp/', '>/var/.ptmx && cd /var/', '>/dev/.ptmx && cd /dev/', '>/mnt/.ptmx && cd /mnt/',</td><td>Mirai variant</td><td>37</td></tr>
<tr><td>7</td><td>(9)</td><td>19</td><td>'>/tmp/.ptmx && cd /tmp/', '>/var/.ptmx && cd /var/', '>/dev/.ptmx && cd /dev/', '>/mnt/.ptmx && cd /mnt/',</td><td>Bashlite variant</td><td>9</td></tr>
<tr><td>8</td><td>3665</td><td>24</td><td>>/dev/.t && cd /dev/;>pppd, >/var/tmp/.t && cd /var/tmp/;>pppd,</td><td>Bashlite variant</td><td>1</td></tr>
<tr><td>9</td><td>(69)</td><td>17</td><td>/bin/busybox ECCHI, /bin/busybox ps;</td><td>Mirai</td><td>69</td></tr>
<tr><td>10</td><td>(3374)</td><td>16</td><td>/bin/busybox ECCHI, /bin/busybox ps;</td><td>Mirai</td><td>3374</td></tr>
<tr><td>11</td><td>823</td><td>18</td><td>/bin/busybox ECCHI, /bin/busybox ps;</td><td>Mirai</td><td>1</td></tr>
<tr><td>12</td><td>(4)</td><td>15</td><td>wget [url]l -O - > dvrHelper; chmod 777 ; /bin/busybox ECCHI</td><td>Mirai</td><td>4</td></tr>
<tr><td>13</td><td>(151)</td><td>13</td><td>(dd bs=52 count=1 if=.s || cat .s), (dd bs=52 count=1 if=.s || cat .s),</td><td>Hajime</td><td>151</td></tr>
<tr><td>14</td><td>2236</td><td>14</td><td>(dd bs=52 count=1 if=.s || cat .s), (dd bs=52 count=1 if=.s || cat .s),</td><td>Hajime</td><td>1</td></tr>
<tr><td>15</td><td>(2)</td><td>12</td><td>(dd bs=52 count=1 if=.s || cat .s), (dd bs=52 count=1 if=.s || cat .s),</td><td>Hajime</td><td>2</td></tr>
<tr><td>16</td><td>4080</td><td>11</td><td>cd /tmp || cd /var/run || cd /dev/shm || cd /mnt || cd /var</td><td>Bashlite</td><td>1</td></tr>
<tr><td>17</td><td>3017</td><td>10</td><td>cd /tmp || cd /var/run || cd /dev/shm || cd /mnt || cd /var</td><td>Bashlite</td><td>1</td></tr>
<tr><td>18</td><td>(4)</td><td>9</td><td>cd /tmp || cd /var/run || cd /mnt || cd /root || cd /, wget/tftp/get/ftpget, chmod, sh, rm</td><td>Bashlite</td><td>4</td></tr>
<tr><td>19</td><td>916</td><td>27</td><td>busybox echo || echo nameserver 8.8.8.8 > /etc/resolv.conf, cd /var || cd /tmp || cd /var/run || cd /var/tmp || cd /dev || cd /dev/shm || cd /mnt || cd /boot || cd /usr || cd /dev/netslink</td><td>Bashlite</td><td>1</td></tr>
<tr><td>20</td><td>(3)</td><td>8</td><td>sh || bash || shell, cd /tmp || cd /var/run || cd /dev/shm || cd /mnt || cd /var</td><td>Bashlite</td><td>3</td></tr>
<tr><td>21</td><td>(2)</td><td>6</td><td>cd /tmp; wget || curl -O ; chmod 777 ; sh</td><td>Bashlite</td><td>2</td></tr>
<tr><td>22</td><td>3703</td><td>7</td><td>cd /tmp; wget || curl -O ; chmod 777 ; sh</td><td>Bashlite</td><td>1</td></tr>
<tr><td>23</td><td>(3)</td><td>3</td><td>sh, shell, help, busybox, wget</td><td>Mirai</td><td>3</td></tr>
<tr><td>24</td><td>2520</td><td>4</td><td>sh, shell, help, busybox, wget/ tftp ; chmod 777 ; sh || bash ; rm -rf ;</td><td>Mirai</td><td>1</td></tr>
<tr><td>25</td><td>2896</td><td>5</td><td>sh, shell, help, busybox, wget/ tftp ; chmod 777 ; sh || bash ; rm -rf ;</td><td>Mirai</td><td>1</td></tr>
<tr><td>26</td><td>(5)</td><td>2</td><td>chmod a+x 7up;./7up., system., /bin/busybox Mirai.</td><td>Mirai variant</td><td>5</td></tr>
<tr><td>27</td><td>726</td><td>28</td><td>cd /tmp, wget/tftp/get/ftpget, chmod, sh, rm</td><td>Bashlite</td><td>1</td></tr>
<tr><td>28</td><td>3921</td><td>29</td><td>busybox wget || wget [url] || tftp [url]; sh</td><td>Bashlite</td><td>1</td></tr>
<tr><td>29</td><td>(934)</td><td>1</td><td>killall wget; killall ping; killall sh while true; do wget -O /dev/null [url] > /dev/null 2></td><td>Unknow</td><td>934</td></tr>
<tr><td>30</td><td>325</td><td>30</td><td>/bin/busybox MIRAI., cd /var/tmp ;cd /tmp; rm -f *; ftpget, sh &</td><td>Mirai</td><td>1</td></tr>
</table>

Table A· 2 The MD5 of malware Tsunami (April 2017, Feature ID 3 of Table A· 1).

MD5
10f045aa890077adc300ea79686eefba
1920b61e9c1e001e2c651bc9ffd59b1a
196dfd58285222b20d6d3434645114e2
25060f2d2d53e80bf01e77ccbabab077
305f120d4893c293faeb368c31ab0913
4312ad5c366a7e500d23883617db8ead
4e9f282659dcc1cd3e9aa9df69d1f9ae
55e1a814fc007a7ac145d8a1f112da9e
571c52660cb9f6f0f3f17f25e871251f
6fae1bce8953e9e16b0fd12361690d23
6ff6033745023abb23277df7de2ae69b
71b3589cd99aa176abd68f647d69bbe7
cd3e728914ba6917911423893d95a75a
cea45d9ad8b5339dbc34bb9d072785f0
d78438bcc89b4ecb62bc089ee4691cb0
d8d5807d12eb2acbdff7eb893b87364f
db2e7c2234302e9db0c47b0891104074
dd900a26248a1d01a3b0cf1c65e8bc44
dd90f88f4028e710da9649d370fda93b
de2569ffa1765d9203397b6b7728358b
f1599964e69bc3d0639a47b38badcdf6
f2f49696f944e793daf73e4de9c67c54
f6c3613139f0f36fa93e88c0ecc13a25

Chun-Jung Wu received his B.S. in mathematics from National Taiwan Uni- versity, Taiwan in 2003 and M.S. in computer Science from National Taiwan University of Science and Technology, Taiwan in 2007. From 2008 to 2016, he was an engineer at Institute for In- formation Industry, Taiwan. Currently, he is a doctoral student at the Graduate School of Environment and Information Sciences, Yokohama National University. His research interest is IoT Security.

Shin-Ying Huang is a R&D manager in Cybersecurity Technology Institute, Institute for Information Industry in Taiwan since 2016. She received her Ph.D. in the Department of Management Information Systems, National Chengchi University, Taiwan. She was a visiting scholar in the Department of Management Information System, University of Arizona (2015–2016). She was a postdoctoral fellow of Research Center for Information Technology Innovation, Academia Sinica (2014–2015). Her research interests are machine learning, AI, cyber security, social media analysis, and threat intelligence. She has been participated in many cybersecurity projects including spam mail management and analysis system, health domain information sharing and analysis center (ISAC) and is currently interested in threat intelligence research in dark web and IoT domain.

Katsunari Yoshioka received his B.E., M.E. and Ph.D. degrees in Computer Engineering from Yokohama National University in 2000, 2002, and 2005, respectively. From 2005 to 2007, he was a Researcher at the National Institute of Information and Communications Technology, Japan. Currently, he is an Associate Professor at the Graduate School of Environment and Information Sciences, Yokohama National University. His research interest covers wide range of information security, including malware analysis, network monitoring, intrusion detection, etc. He was awarded 2007 Prizes for Science and Technology by The Commendation for Science and Technology by the Minister of Education, Culture, Science and Technology.

Tsutomu Matsumoto is a professor of Faculty of Environment and Information Sciences, Yokohama National University and directing the Research Unit for Information and Physical Security at the Institute of Advanced Sciences. He received Doctor of Engineering from the University of Tokyo in 1986. Starting from Cryptography in the early 80's, he has opened up the field of security measuring for logical and physical security mechanisms. Currently he is interested in research and education of Embedded Security Systems such as IoT Devices, Network Appliances, Mobile Terminals, In-vehicle Networks, Biometrics, Artifact-metrics, and Instrumentation Security. He is serving as the chair of the IEICE Technical Committee on Hardware Security, the Japanese National Body for ISO/TC68 (Financial Services), and the Cryptography Research and Evaluation Committees (CRYPTREC) and as an associate member of the Science Council of Japan (SCJ). He was a director of the International Association for Cryptologic Research (IACR) and the chair of the IEICE Technical Committee on Information Security. He received the IEICE Achievement Award, the DoCoMo Mobile Science Award, the Culture of Information Security Award, the MEXT Prize for Science and Technology, and the Fuji Sankei Business Eye Award.