

Counterfeit Fingerprint Detection of Outbound HTTP Traffic with Recurrent Neural Network

Chi-Kuan Chiu*, Te-En Wei*[†], Hsiao-Hsien Chang*, Ching-Hao Mao*

*CyberTrust Technology Institute, Institute for Information Industry, Taiwan, R.O.C

[†]Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology
tonychiu@iii.org.tw, dwyanewei@iii.org.tw, arieschang@iii.org.tw, chmao@iii.org.tw

Abstract—Most of malwares usually hide their malicious activities into HTTP protocol, such as communicating with the C&C server or accessing some malicious webpages. The previous detection system is mainly based on the blacklist. With the advancement of technology, there are many ways can easily available to evade detection system nowadays, e.g., spoofing the HTTP headers. However, the stealthiest malware still needs to communicate with the destination point. This paper aims to provide a detection system to detect these stealthily malicious activities. Consider machine-learning classifiers with manually handled features have been widely used in malicious URL detection. We propose an end-to-end deep learning framework to learn the URL embedding for application classifier from URL string. We apply the Recurrent Neural Network (RNN) to effectively keep the semantic meaning and sequential patterns in URL strings. The proposed approach is evaluated with real-world data from technology enterprise's network and compare the performance with the state-of-the-art approach. The result shows that our approach achieves the accuracy rate to 99%, and even the counterfeit fingerprint's detection reaches up to 100% while the comparison approach failed with the same scenario.

Index Terms—Anomaly Detection, Application Fingerprinting, Network Security

I. INTRODUCTION

Nowadays, malware usually uses HTTP protocol to connect suspicious host for data leakage and exfiltration, because it's a common network channel that Intrusion Detection/Prevention Systems (IDS/IPS) never block the HTTP traffics. Therefore, the malware tries to hide their penetrations in the HTTP traffic to evade the detection. The proposed method that uses the fingerprint to detect malware hide in outbound HTTP traffics [2], and which can't efficiently detect malware when hacker generates counterfeit fingerprints.

The Fingerprints are easily modified by hackers. Therefore, Chi-Kuan and Te-En et al. [3] use graph edit distance with the HTTP fields (domain and referer). They calculated the distance between domain and referer to detect the requests in the browser whether spoofing or not. It is not a suitable method that analyzing the traffic log of applications in HTTP headers. In general, the information of referer is not contained in the logs of applications. Thus, we represent the problem defined as follows:

• Problem Definition

Malware could hide the communication with the fingerprint spoofing (e.g., user-agent, accept language, and so on) in the HTTP header to evade intrusion detections. The

destination domain field in HTTP header usually is fixed which is an identification string represents the destination of the hostname, mapping with the IP address. Therefore, the destination domain would be changed when malware connects C&C server even using fake HTTP header.

With this problem, we represent an approach based on URL classification with Machine Learning in training and testing phases, and the contributions of this work are briefly summarized as followings:

- The solution that we proposed would detect outgoing anomalous HTTP traffic, which is based on application fingerprint and URL classification techniques. Furthermore, our approach automatically generates fingerprints from network traffic, filter anomalous communications from the monitored hosts, and then identify counterfeit fingerprint in normal traffics which are filtered by fingerprints [2]
- We apply a counterfeit fingerprint detection to resolve the counterfeit problem. The detection system would identify fake contents of HTTP header based on URL classification. Besides, we also have implemented a system based on our approach in Python, and the current state of the art regarding client-side anomaly detection which is already in collaboration with other institution and enterprise.
- Our detection system would be evaluated and compared with DECANter [2] through real-world datasets. In the performance results, it shows that our approach is better, especially the counterfeit fingerprint is hard to evade from our approach.

In the remaining parts of this report, Section 2 describes the detail components of the proposed approach. The effectiveness, performance, and case studies of the proposed framework are evaluated and discussed in Section 3. Section 4 surveys related work. At last, Section 5 concludes this project.

II. PROPOSED APPROACH

This section would go into detail about our proposed approach which go for detecting counterfeit fingerprints from applications' outbound HTTP traffics.

A. System Overview

Before going further, all PCAP files collected by an enterprise's host. The network activities generated by a set of applications such as $A = a_1, \dots, a_n$, and which are installed in

TABLE I: Fields and Values of Database in a PCAP File

Field	Value for Instance
<i>Domain</i>	cr-tools.clients.google.com
<i>User-agent</i>	Google Update/1.3.21.65;winhttp
<i>URI</i>	/service/check2?appid= ...
<i>Length</i>	433
<i>Const field</i>	'Connection', 'Additional-Headers', 'Pragma'

the enterprise's hosts. Each application a_i has several PCAP files which contain specific network characteristics, and our proposed approach possibly create a fingerprint f_{a_i} for each application.

The PCAP files of a host H include union of all application fingerprints which is defined as $H = \cup_j^n f_{a_i}$. The proposed counterfeit fingerprint detection process consists of training and testing phases. In the training phase, we assume enterprise hosts aren't compromised.

This method is composed by three components. In the training phase, it would take the PCAP file as input and classifies traffics, and then construct fingerprints and generate URL classifier which calls Header Inspector. In the testing phase, given an application outbound HTTP traffic reconstructed by fingerprint, and the second step filters benign browser traffics through fingerprint matching. Continuously, process the URL strings and compare the URL classifier output with the UserAgent of the request's header field for counterfeit fingerprint detection. The proposed method is depicted in figure 1 and the following paragraphs describe the details of each component.

B. Traffic Extractor

For most cases of client-side attacking, hackers whose general goal is to steal valuable data before malware connects to C&C server. As a result, PCAP files, that contain specific network characteristics of an application for each host in the enterprise. To generate the fingerprint for each application, our approach first extracts various entities from PCAP files. Table I shows 5 heterogeneous fields which can be extracted from each one-line log, including domain (*Domain*), user-agent (*User-agent*), URI (*Uri*), length (*Length*), and const fields (*Const field*). The reason for choosing these 5 fields for application traffic classification can be summarized as followings and fingerprint construction is represented in the next subsection.

In previous research [2], Bortolameotti et al. identified two types of HTTP applications (e.g., *browser* and *background*). This subsection aims to filter logs of a PCAP file according to the *User-agent* because we focus on counterfeit fingerprints of application network activities. To identify application activities, the browser flags we defined are "Firefox", "Chrome", "Safari", "OPR", "Opera", "MSIE", "Gecko", "Trident", and "AppleWebKit", and which are used for string matching in field *User-agent*. We can filter out the browser traffic with this approach.

C. Fingerprint Module

Single feature (e.g., *User-agent*) isn't effective enough to filter normal network activities [2] [4]. Therefore, we consider

multiple features such as *Domain*, *URI*, *Length*, and *Const field* for fingerprint generation, and *Domain* and *URI* would be used for sequentially URLs-based classifier in other subsection. In our assumption, hacker will not be so lucky to guess all parameters of *User-agent*, *Length*, and *Const field* at the same time. In this subsection, we denote a set of *User-agent* $U = u_1, \dots, u_n$, a set of *Domain* $D = d_1, \dots, d_q$, and a set of *Const field* $C = c_1, \dots, c_m$ where $|U| = n$, $|D| = q$, and $|C| = m$. Furthermore, our approach makes fingerprint $f = (u_i, c_j, d_k)$ where $i = 1 \sim n$, $j = 1 \sim m$, $k = 1 \sim q$, and $|f| = n \times m \times q$. Matching testing application fingerprint to knowns which is trained and stored in database, and we would briefly show the similarity estimation in following module.

Matching fingerprint we used is an easy comparison in this module [2]. In training phase, we take fingerprints f_{a_i} for each application a_i . If our approach constantly runs in testing mode, we must obtain other application a_j fingerprints f_{a_j} .

D. Header Inspector

In previous research [2], they observed that most of the applications that run as background services often communicate with the same set of domains. Therefore, we define the Header Inspector classify a given URL to the correspondent application represented by UserAgent. Furthermore, it will be combined with the result got from Fingerprint module which detects the suspicious request.

There are three main parts in Header Inspector: Tokenizer, Classifier, and Compare Module. Tokenizer responsible for dividing the URL into the hostname, path, and parameter keys as the figure 2. And it turns strings into sequences with the fixed pad. In the next step, the classifier receives the input from Tokenizer and predict the result as of UserAgent. It would cost lots of time for training and less time for predicting. The compare module would compare the predicted result from the classifier with the real one which is in HTTP. With the Fingerprint module, our system can greatly reduce the false negatives caused by fingerprint technology.

With the URLs features transforming, we can treat the tasks as the NLP tasks. We would consider the meanings of nearby characters in the URL. The patterns of sequential will help us to improve the performance of our system. Therefore, our system chooses to use RNN layers to extract features and using the fully connected layer for classification tasks.

As the Figure 3, to handle our outbound HTTP traffic, we use Fingerprint module to extract and compare the requests. The Header Inspector analyzes the benign data from the Fingerprint module's output. The Tokenizer would process the URLs to the word vector and analyzed by the classifier. As the inputs of the classifier are sequential, we chose to use Recurrent Neural Networks(RNNs) to be able to capture the sequential transform features.

III. EXPERIMENT RESULTS

In this section, we would describe the datasets that we used to perform our experiments. For the experiments, we have collected two types of datasets, simulated and real-world data.

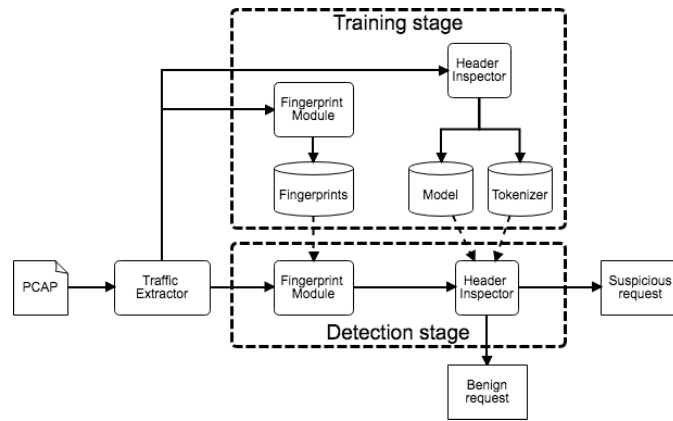


Fig. 1: A overview of our counterfeit fingerprint detection system. This system mainly consists of two phases with training and detection stage.



Fig. 2: The features of URLs

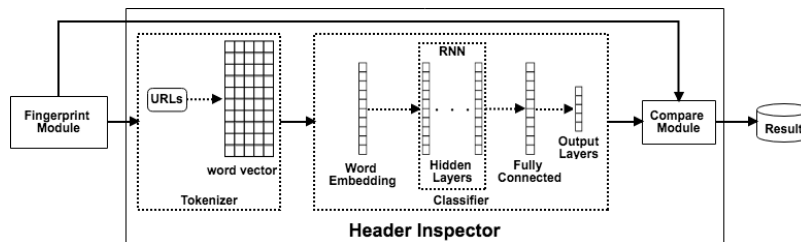


Fig. 3: The detail of the Header Inspector

We combined the datasets that be able to evaluate the detection performance of our system or DECANter [2].

A. Experimental Settings

In the following, we briefly present the datasets we used for evaluation in our system. The dataset information is represented in table II.

• Real-world Data

The outbound HTTP traffics we collect from more than hundreds of machines in a technology industry. Users of these machines include accountants, engineers, sales executive, and administrative personnel. Since the users vary from different occupations that make the data become more various and complexity. We split the real world dataset into two sets, one is training and the other is testing. Training set has covered the first few days and the testing set is the traffics of last days. In training set, Industry flow_01 contains 1,690,869 HTTP requests. and the testing set Industry flow_02 contains 68,234 HTTP requests in this real-world dataset.

• Simulated Data

In this paper, we present a technique for detecting the counterfeit fingerprint which would pretend to be application activities in outbound HTTP traffics. However, this

kind of attack is secret and hidden penetration, and hard to collect in the real world. Therefore, we build a botnet malware and monitor its outbound HTTP traffics. As we know, early botnets generally used Internet Relay Chat (IRC) channel to communicate C&C server. In recent years, botnets also start to communicate C&C server through HTTP protocol. Furthermore, we need to build a botnet with the spoofing headers which can evade other detection systems. That is why we collect three different simulated botnet traffics. In Dataset_01, it has no spoofing headers from infected host's outbound HTTP traffics. Dataset_02 is totally modifying the headers information, and the malware investigates which application is used by the user or host, and fills the field User-Agent with that specific application. Moreover, it also fills up some common requests header fields, like Accept, Accept-Encoding, and Const field.

B. Evaluation Metrics

Essentially, Our system is a flow filter aim to identify the suspicious requests with the headers field. Four well-known metrics for evaluating the effectiveness of the proposed method are adopted as followings: "true positive"(TP) means the number of normal requests which belong to normal traffic.

TABLE II: Overview of the Datasets.

Dataset	Features	Types	All Samples	Benign Samples	Malicious Samples
Industry_flow01	Packets	Traffic Flows	1,690,869	1,690,869	N/A
Industry_flow02	Packets	Traffic Flows	68,234	68,234	N/A
Dataset_01	Packets	Botnet	1,201	1,039	162
Dataset_02	Packets	Botnet	11,642	11,420	222

“False negative”(FN) is the number of normal traffic and its results are wrongly predicted. Similarly, “true negative”(TN) means the number of abnormal traffic and the system predict it as malicious requests, while “false positive”(FP) is the number of abnormal traffic that the system predicts it as normal traffic. Based on the accumulation of TP , FN , TN , and FP , one extended metrics (*accuracy*) popularly used in machine learning problems are also adopted here to evaluate proposed method and listed in equations below. Note that the optimal *accuracy* of 1.0 means all of the malware are successfully picked out by the proposed approach.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

C. Effectiveness Analysis

Just as we know, malware can easily modify the HTTP header, therefore, we build three kinds of malware to evaluate our approach. With these botnets, they all have the same purpose, and which would steal some sensitive information (such as OS information, system account, and the password) and send requests to the C&C server periodically waiting for commands to execute. The botnet in dataset_01 only simply sets the User-agent as a common application in the Windows operation system which calls Microsoft BITS. In the Dataset_02, the attacker would analyze the Microsoft BITS application and set the botnet’s HTTP headers to disguise as the application. For testing system’s robustness, we merge dataset Inudstry_flow 02 to each simulated data. It may greatly increase the complexity of the detection task. The result is shown in table III, we can see that our system and DECANter [2] both have the great performance with Dataset_01. However, we can notice that our system has a better performance for botnets in Dataset_02 based on advanced spoofing methods. We can detect the traffics that have suspected counterfeit HTTP header fields more effectively by using the header inspector.

In the training stage, We build each fingerprint through the clean data for both systems. For with no User-Agent request headers, which filled in '-' in User-Agent field, we will use the domain and IP to create the fingerprints. Moreover, this concept is just like the idea of the whitelist, and malware can’t evade our system detection through the settings of HTTP header with empty User-Agent. The result shows that two detection systems both got an excellent performance in the Dataset_01. Some malware would use an application-like User-Agent to camouflage themselves as an application to avoid the system detection. In this validation set, the malware disguises itself as a normal application, and our system would

firstly check the request whether whitelist has this fingerprint or not. If yes, the system will compare the difference with these fingerprints. So, unless the malware can guess or use some advanced technology to analyze the application information, and then using this information to forge the HTTP header fields. Otherwise, it is not easy to avoid the first stage of our detection system.

For advanced spoofing methods, the malware can use some advanced technology to analyze the application information, and modify the HTTP header fields when sending requests to C&C server. In Dataset_02, the malware would modify the User-Agent with Microsoft BITS, and disguise itself as a normal application to communicate with C&C server. For this reason, it is not robust with only the first phase of detection (e.g., DECANter [2], DUMONT [5], and WebTap [6]). Therefore we also established the Header Inspector during the training phase. For the traffic that has passed the first phase with the detection system, we would execute the second phase of detection with Header Inspector module. The idea of this detection stems from the fact that we think the malware can easily modify the HTTP header fields, but it can not change the destination field. We apply Header Inspector to analyze the URLs and compare the User-Agent between the predict and real one. The results show that our approach is better than DECANter [2], and it could detect the malware sending requests in Dataset_02 which using advanced spoofing methods to evading detection system.

D. Limitation and Future Work

Our approach has a limitation in practice, the problem is the duration of model training or fingerprint constructing that need a non-compromised environment. Because our approach is an extension of DECANter [2], and which needs to generate an application’s fingerprint for anomaly detection in a clean environment. Therefore, we also have to perform our approach at the same scenario for benign fingerprint and Header Inspector.

We only train our model in the cooperative enterprise, but the difference of the application’s network activities would be not the same from each different applications. It would intensely increase the false positive and false negative rate. Therefore, we will design a system to combine the methods of supervised and unsupervised learning to figure out the unknown application’s problem in the future.

IV. RELATED WORK

Many studies [7] [8] [9] [10] focus on the C&C connection detection for identifying Botnet, such as: IRC, HTTP, SMTP in the phases of probe, penetrate, escalate, expand of cyber

TABLE III: Overview of Evaluation. For testing the robustness, we add the real-world traffic into each simulated data. We compare the performance of our system and DECANter [2] against different data sets. The accuracy is calculated by equation (1) and the detail result will show below.

Dataset	System	HTTP Requests	Evaluation Metrics				Accuracy	Counterfeit Fingerprint Detection Rate
			TP	TN	FP	FN		
Dataset_01+Industy_flow02	DECANter [2]	69,435	69,273	162	0	0	1.0000	N/A
	Our System		69,272	162	0	1	0.9999	N/A
Dataset_02+Industy_flow02	DECANter [2]	79,876	78,333	0	222	1321	0.9806	0.0000
	Our System		79,653	222	0	1	0.9999	1.0000

killer chain. And Livadas et al. [11] [12] represent a machine learning based approach to detect botnet by using the network traffic. The approach is focused on chat-like protocols such as IRC. Malware clustering is capable to generate the malware signatures of communication on the network. Several studies generate the signatures via clustering approaches from known botnet traffics [13] [14]. Perdisci et al. [15] [16] proposed a technique that using URL similarities, and extracts sub-tokens from URLs to clusters malware sample. And using the cluster to identify malicious traffic on the network. Rafique and Caballero [17] proposed a tool which calls FIRMA that generates a set of signatures for each malware family by aggregates malware samples into families based on similar protocol features.

Anomaly-based detections leverage different kinds features to identify variant malware traffics such as detail of service [18], context of Web attacks [19], and encrypted data exfiltration [20]. Most works need a large number of samples to train the models [22] [23], in addition, several works incorporates network and host features to design the detection [24] [6]. To solve false positive of malicious samples, some studies [25] have the mixed real-world data with simulated data. Furthermore, Eslahi et al. [26] proposed an approach to reduce the false positive rate in HTTP-based botnet detection. However, most kinds of these works still suffer high false positive problems result in being impractical.

Many studies use the HTTP headers to detect suspicious traffic is mainly focused on the User-Agent. As Kheir et al. work [27], they use the general signatures which call of User-Agents to filter out the same signature cluster as labeled as legitimate and the left is labeled as malicious. The biggest problem with this approach is that it can only detect specific malware with used constant User-Agent field to connect or send through HTTP protocol. And the report [28] shows that it is not only used in web browsing but also other types of applications like variant malware attacks. In 2017, Bortolameotti et al. [2] proposed DECANter system to identify the outbound HTTP traffics whether anomalous or not. The proposed method not only intends to create a detection model without from sets of known malware samples but also without additional knowledge of threats or known malware samples. Therefore, we try to fix this problem in this paper.

V. CONCLUSION

We propose an anomaly detection for HTTP-based applications based on fingerprint and the destination of URLs.

Our system doesn't need any samples of malware activities while training the classifier or generating the fingerprints. It could figure out the bias issue that the detection system usual fit in specific malware. Furthermore, we also could deal with the advanced of header spoofing by malware in our approach. Most of the existing solutions focus on modeling the statistical value of network traffic or even analyze network payload to identify specific malware in usual. Our system would adopt innovative inspection technique by integrating the fingerprint and sequentially URLs-based classifier to passively detect the malware. We also compare to existing system DECANter [2], and which shows our system has a better detection performance result. Contributions of our work can be summarized as followings:

- **Well Problem Modeling** We collect the real-world data of network traffic and define the computable problem of learning client-side intrusion threats through the well-defined adjacent vector and the sequentially URLs-based classifier.
- **Automatic and Accurate Threat Detection** Our system considers the relationships of the heterogeneous entities such as user-agent, domain, URI, length, and const field at the same time. And the different attacking scenarios would show various relationships among those entities. We design the system that contains the adjacent vector and sequentially URLs-based classifier would take the complementary detections. The result shows that our system with the accuracy (99%) and recall (99%) perform better than the previous approach, especially in the advanced covert attacking threat with the counterfeit fingerprint.

ACKNOWLEDGMENT

This research is supported by industry and institute in Taiwan, and the authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the contributions of this paper.

REFERENCES

- [1] G. Gu, J. Zhang, and W. Lee, "Botsniffer: Detecting botnet command and control channels in network traffic," 2008.
- [2] R. Bortolameotti, T. van Ede, M. Caselli, M. H. Everts, P. Hartel, R. Hofstede, W. Jonker, and A. Peter, "Decanter: Detection of anomalous outbound http traffic by passive application fingerprinting," in *Proceedings of the 33rd Annual Computer Security Applications Conference*. ACM, 2017, pp. 373–386.

- [3] H.-H. C. Chi-Kuan Chiu, Te-En Wei and C.-H. Mao, "Counterfeit fingerprint detection of outbound http traffic with graph edit distance," in *The 2018 IEEE Conference on Dependable and Secure Computing*, IEEE, 2018.
- [4] N. Kheir, "Analyzing http user agent anomalies for malware detection," in *Data Privacy Management and Autonomous Spontaneous Security*, Springer, 2013, pp. 187–200.
- [5] G. Schwenk and K. Rieck, "Adaptive detection of covert communication in http requests," in *Computer Network Defense (EC2ND), 2011 Seventh European Conference on*, IEEE, 2011, pp. 25–32.
- [6] K. Borders and A. Prakash, "Web tap: detecting covert web traffic," in *Proceedings of the 11th ACM conference on Computer and communications security*, ACM, 2004, pp. 110–120.
- [7] M. Feily, A. Shahrestani, and S. Ramadass, "A survey of botnet and botnet detection," in *Emerging Security Information, Systems and Technologies, 2009. SECUREWARE'09. Third International Conference on*, IEEE, 2009, pp. 268–273.
- [8] P. Barford and V. Yegneswaran, "An inside look at botnets," in *Malware detection*, Springer, 2007, pp. 171–191.
- [9] W. T. Strayer, D. Lapsley, R. Walsh, and C. Livadas, "Botnet detection based on network behavior," in *Botnet Detection*, Springer, 2008, pp. 1–24.
- [10] A. C. Atluri and V. Tran, "Botnets threat analysis and detection," in *Information Security Practices*, Springer, 2017, pp. 7–28.
- [11] L. Carl *et al.*, "Using machine learning techniques to identify botnet traffic," in *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, IEEE, 2006.
- [12] W. T. Strayer, R. Walsh, C. Livadas, and D. Lapsley, "Detecting botnets with tight command and control," in *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, IEEE, 2006, pp. 195–202.
- [13] A. Zarras, A. Papadogiannakis, R. Gawlik, and T. Holz, "Automated generation of models for fast and precise detection of http-based malware," in *Privacy, Security and Trust (PST), 2014 Twelfth Annual International Conference on*, IEEE, 2014, pp. 249–256.
- [14] T. Nelms, R. Perdisci, and M. Ahamad, "Execscent: Mining for new c&c domains in live networks with adaptive control protocol templates," in *USENIX Security Symposium*, 2013, pp. 589–604.
- [15] R. Perdisci, D. Ariu, and G. Giacinto, "Scalable fine-grained behavioral clustering of http-based malware," *Computer Networks*, vol. 57, no. 2, pp. 487–500, 2013.
- [16] R. Perdisci, W. Lee, and N. Feamster, "Behavioral clustering of http-based malware and signature generation using malicious network traces," in *NSDI*, vol. 10, 2010, p. 14.
- [17] M. Z. Rafique and J. Caballero, "Firma: Malware clustering and network signature generation with mixed network behaviors," in *International Workshop on Recent Advances in Intrusion Detection*, Springer, 2013, pp. 144–163.
- [18] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to ddos attack detection and response," in *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, vol. 1, IEEE, 2003, pp. 303–314.
- [19] C. Kruegel and G. Vigna, "Anomaly detection of web-based attacks," in *Proceedings of the 10th ACM conference on Computer and communications security*, ACM, 2003, pp. 251–261.
- [20] G. He, T. Zhang, Y. Ma, and B. Xu, "A novel method to detect encrypted data exfiltration," in *Advanced Cloud and Big Data (CBD), 2014 Second International Conference on*, IEEE, 2014, pp. 240–246.
- [21] H. Zhang, D. D. Yao, N. Ramakrishnan, and Z. Zhang, "Causality reasoning about network events for detecting stealthy malware activities," *computers & security*, vol. 58, pp. 180–198, 2016.
- [22] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "Disclosure: detecting botnet command and control servers through large-scale netflow analysis," in *Proceedings of the 28th Annual Computer Security Applications Conference*, ACM, 2012, pp. 129–138.
- [23] G. Gu, R. Perdisci, J. Zhang, W. Lee *et al.*, "Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection," in *USENIX security symposium*, vol. 5, no. 2, 2008, pp. 139–154.
- [24] G. Schwenk and K. Rieck, "Adaptive detection of covert communication in http requests," in *Computer Network Defense (EC2ND), 2011 Seventh European Conference on*, IEEE, 2011, pp. 25–32.
- [25] W. Wang, Y. Wang, X. Tan, Y. Liu, and S. Yang, "Botcapturer: Detecting botnets based on two-layered analysis with graph anomaly detection and network traffic clustering," *International Journal of Performability Engineering*, vol. 14, no. 5, p. 1050, 2018.
- [26] M. Eslahi, H. Hashim, and N. Tahir, "An efficient false alarm reduction approach in http-based botnet detection," in *Computers & Informatics (ISCI), 2013 IEEE Symposium on*, IEEE, 2013, pp. 201–205.
- [27] N. Kheir, "Behavioral classification and detection of malware through http user agent anomalies," *Journal of Information Security and Applications*, vol. 18, no. 1, pp. 2–13, 2013.
- [28] K. Bartos, M. Sofka, and V. Franc, "Optimized invariant representation of network traffic for detecting unseen malware variants," in *USENIX Security Symposium*, 2016, pp. 807–822.



Chi-Kuan Chiu received his B.S. degree in applied mathematics from Chung Yuan Christian University, Taoyuan, Taiwan, R.O.C, in 2013. He received the M.S. degree in international business from Chung Yuan Christian University, Taoyuan, Taiwan, R.O.C, in 2015. He is currently an engineer in Cybersecurity Technology Institute, Institute for information industry, Taiwan, R.O.C. His research interests include machine learning, deep learning, computer vision, image processing and security.



Te-En Wei received the M.S. degree in electrical engineering from National Taiwan University of Science and Technology, Taipei, Taiwan, R.O.C, in 2007. He is currently a data scientist/ section manager in Cybersecurity Technology Institute, Institute for information industry, Taiwan, R.O.C. His research interests include machine learning, data mining, graph mining, deep learning, generative adversarial network.



Hsiao-Hsien Chang received the B.S. degree in computer science and information engineering from Fu Jen Catholic University, Taiwan, R.O.C, in 1995. He received the M.S. degree in computer science and information engineering from National Chung-Hsing University, Taiwan, R.O.C, in 1997. He is currently a deputy director in Cybersecurity Technology Institute, Institute for information industry, Taiwan, R.O.C. His research interests include project management, customer relationship management, system design and development and machine learning.



Ching-Hao Mao received the B.S. degree in information management from Tamkang University, Taipei City, Taiwan, in 2002. He received the M.S. degree in information management from Chinese Culture University, Taipei, Taiwan, R.O.C in 2004. He received the Ph.D. degree in Computer Science and Information Engineering from National Taiwan University of Science and Technology, Taipei, Taiwan, R.O.C in 2010. He is currently a vice president and director general in Cybersecurity Technology Institute, Institute for information industry, Taiwan, R.O.C. His research interests include network security, intrusion detection, event analysis, botnet detection, web application security, data mining.