A Method Based on Statistical Characteristics for Detection Malware Requests in Network Traffic

Ke Li[1,2], Rongliang Chen[1], Liang Gu[2], Chaoge Liu[3], Jie Yin[3]
[1](Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences)
[2](Sangfor Inc.)
[3](Institute of Information Engineering, Chinese Academy of Sciences)
like@sangfor.com.cn

***Abstract*—Network traffic inspection is an important method to discover the existence of malware when it bypasses security devices through polymorphic techniques or zero-day attacks. However, traditional network signature-based or IoC (Indicator of Compromise) detection could fail since the encryption and variability of threats has been increasing. It is well known that these methods are fragile and have difficulty dealing with new variants. This paper proposes a system designed to detect security threats based on the statistical characteristics of HTTP requests from malware. The corresponding method does not rely on signatures or the specific command and control (C&C) contents between bots and the botmaster and can be built given easily accessible information extracted from the HTTP data log, including HTTP headers and the URL. The results from millions of live traffic flows show correct detection with a precision exceeding 98.32% for malicious flows, and the recall reaches 98.70%. We believe this detector represents the main mechanism for discovering network threats in the future.**

Keywords—Malware Detection, Network Traffic, Machine Learning

I. INTRODUCTION

Along with the increasing number of severe attacks (e.g., EternalBlue exploit), new and polymorphic malware threatens an increasing number of Internet users and firms. Through common evasion methods such as the domain generation algorithm (DGA) or encryption, the threats are always difficult to detect and block using signature-based security devices. For example, existing malware detection techniques based on the regular expression to capture invariant tokens or patterns on the URL [1][2] become ineffective, and malware could use obfuscation methods [3] such as Base64 encoding to easily avoid pattern matching.

The alternative detection technique, based on statistical features from network packets or flows, enables the discovery of malware behaviors. This technique is more expressive than signature-based approaches and has great generalization abilities. The main point to detect malware behaviors is that the botmaster needs to communicate with infected hosts for manipulation, including launching various Internet attacks such as DDoS, spam, click fraud, and illegal BitCoin mining. The C&C traffic features from the same malware or even malware family may exhibit the same characteristics, which is difficult to completely change using the mentioned evasion strategies. The statistical characteristics-based detection shows the promise to discover the unique behaviors of malware.

In this paper, we propose the MalHunter based on behavioral related statistical characteristics for the network-level detection of malware. Given that the HTTP protocol is the most popular method used by malware and that statistical features between different protocols are very different, the MalHunter mainly focuses on the detection of malware that leverages the HTTP as the primary channel to communicate with the C&C server or to launch attack activities. In contrast to previous works, the approach here does not rely on the C&C contents, which are always encrypted with an XOR or RC4 algorithm. In addition, the method mainly focuses on HTTP requests rather than responses. If the C&C server is temporarily offline or changes its response content, there is little impact on our detection capabilities. The MalHunter is applicable to network traffic analysis (NTA) related security devices to classify HTTP communications captured in a monitored network as benign or malicious and to further identify potential infected hosts and suspicious destinations.

The MalHunter is trained to find malware communication patterns from three types of features that are most important to distinguish between benign and malicious HTTP requests: a) *character distribution of the URL*, which refers to the distribution of each type of character (e.g., alphabet, digital) and the length of each part of the URL (e.g., domain, path, parameters) that could represent the behavior pattern to some degree; b) *HTTP header fields*, which is a useful category of features that some previous works have validated before [4][5][6] and that our research uses for reference; and c) *HTTP header's sequence*, which is the ideal indicator to distinguish different applications (e.g., browser), which uses the N-gram method to represent the unique fingerprint. In general, the MalHunter can identify different types of request flows flexibly and accurately and can be combined with other flow detection techniques, such as Netflow detection.

To demonstrate the feasibility of our design in the real world, we implement the prototype and experimental evaluation of MalHunter using known Trojan or Botnet family samples and real data from volunteer networks. Our results show that MalHunter detects almost all HTTP requests sent by the malware with precision and recall ratios respectively reaching 98.32% and 98.70%.

978-1-5386-4210-8/18/$31.00 ©2018 IEEE
DOI 10.1109/DSC.2018.00084

In summary, this paper presents the following major contributions.

- We propose a new approach to distinguish the traffic generated from HTTP-based malware and benign applications. Different from previous work, our method inspects the statistical features of requests instead of responses, which is robust to some attackers' evasion strategies, such as HTTP body and URL encryption and changes to reply content.
- We introduce a binary classification to detect malicious HTTP traffic using character distributions of URL, HTTP header fields and HTTP header's sequence features. This model overcomes the defects of single dimensional features and is suitable for general malware traffic detection.
- We implement and evaluate MalHunter. The experimental results show it is able to correctly identify malware traffic with high precision and recall on million-scale datasets.

The rest of this work is organized as follows. In Section 2, we present the related work. In Section 3, we introduce the detection approach in details. In Section 4, we present the design of the MalHunter. In Section 5, we discuss the evaluation of MalHunter using real network traffic. In Section 6, we discuss the limitations of our approach. Finally, in Section 7 we draw the conclusion.

II. Related Work

In practical scenarios, different detection strategies are applied to network intrusion detection systems that are deployed at the edge of various local networks to monitor both incoming and outgoing traffic. One of the most common methods is signature-based detection techniques, such as yara or snort rules. There is significant research that focus on how to automatically generate high quality network signatures. Perdisci et al. [1][2] introduced a URL signature automatic generation method that used a two-stage clustering to determine similar malicious flow, the experimental results show that the method has high accuracy and is able to discover subsequent samples from the same family. Zarras et al. [7] believed that the traditional signature generation method is too general and lacks a higher level in its dealings with HTTP botnet detection information. Therefore, they proposed the detection system BOTHOUND, which combined the HTTP header field chains and HTTP signature templates to accurately identify botnet traffic. Rieck et al. proposed the framework BotZilla [8], which generates signatures from tokenized substrings of several proxy log strings, including the user agent and URL. They define signatures as substrings that occur at least d times in a training set and were not seen more than n times in legitimate traffic.

Some detection systems rely on passive DNS analysis to detect malicious domains. The Exposure system [9] monitors long time periods and extracts distinct DNS statistics to train a C4.5 decision tree to detect a malicious domain. Sharifnya et al. [10] proposed a botnet reputation system to find domain-flux botnets, which checks whether the host has abnormal behavior in the DNS query, including the character distribution of domain, the spatiotemporal similarity of host requests, a large number of NXDOMAINs, and others. The system can effectively reduce the high false alarm rate of traditional detection methods and is adapted to large-scale network environments. Schiavoni et al. [11] also put forward a DGA domain recognition system. They believe that a DGA domain does not satisfy language rules (i.e., patterns of vowels and consonants), so combined with the unsupervised learning method and mapping relationship between the domain and IP, the output, DGA domain name fingerprint, can be used for the detection and tracking of botnets.

Some other researches prefer to discover malicious traffic through abnormal host-based or network-based behaviors. The BotMiner [12] could identify different botnets based on discovering the host abnormal behavior, and the A-plan is correlated with the clustering of communication patterns from C-plan. Lu and Brooks [13] used the hidden Markov model to distinguish normal traffic and botnet traffic based on the polling feature of HTTP malwares. The experiment proved the method can effectively identify ZeuS botnet traffic. Grill and Rehack [14] found that 95% of malwares use hard-coded user-agent fields and nearly all of them have errors or non-browser values. Based on the above facts, they classify the user-agent values and then use the access behavior statistics to identify malicious HTTP traffic.

Complementary to previous studies, our research has the following features: 1) Our work classifies each HTTP flow and does not rely on external data. Without multi-stage clustering progress, the computational costs are much lower, and the speed is ideal. 2) Compared to signature-based methods, our work need not reverse engineering or perform tedious signature extraction processes.

III. Detection Approaches

A. Threat and Target

The purpose of this study is to detect the HTTP-based threats from a mass of network data without relying on malicious software specific behaviors, such as files, processes, and registry behaviors. Our detection system learns the special HTTP communication patterns from malware and uses the knowledge to detect the presence of compromised machines in the monitored network. To be more precise, we mainly (but not only) discover the C&C channel or malicious network activities over the HTTP.

In general, the C&C servers of botnets tend not to survive long, resulting in a large number of HTTP requests that do not have completed responses. If we consider the statistical features from full sessions, the method will encounter the problem of data loss and affect the accuracy of detection in the model training and practical application. Therefore, our detection is mainly based on request packages and tends to be more robust and covers more types of malicious traffic.

After our long observation of traffic patterns from different malwares, we manually designed features from multiple dimensions to represent the differences between legal flow and malicious flow to some degree. It is important to note that each

characteristic is weak, but the overall binary classification based on hundreds of these features can obtain a good effect. In the rest of this section, we describe the three strategies of feature selection in detail.

B. URL Statistical Features

As the expression of network resource locations and the access method, the URL played an important role in detection for Web attacks, such as identifying malicious Web pages and DGA domains [11][15]. Our method also introduces related characteristics for detection.

We first extract the character-level features of the entire URL and capture statistical properties in each subsection (See Fig. 1). The characteristics are divided into the following aspects: a) Features of URL components; characterize the distinct patterns in the URL string, including the length of the URL, the vowel ratio, the consonant ratio, the number of special characters ('!', '-', '_', '.', ',', '@', '#', '%', '+', ':', ';'), the uppercase ratio, the lowercase ratio, and the proportion of the digits. b) Domain; common statistical features of domain names, including domain name levels, character type distribution ratio, and top-level domain name. c) Path; this section selects the overall length of the path and the number of layers. d) File name; considering that some malwares have special characteristics of files in the call home or download phases (e.g., no suffix, .php), we select file suffixes and the filename length as features. e) Parameters; in addition to the usual features, such as the number of parameters, the average length of the parameter values, and whether Base64 encryption is used, we also check whether the parameters conform to the ordinary pattern, namely, whether '=' and '&' are repeated in the specified order [16].

As mentioned above, we have extracted a total of 29 dimensional features. However, only introducing these features could not fully identify malicious traffic. For example, attackers may attempt to encode the path and parameters, or use a common file path to cheat the classifier. To achieve more accurate detection, we need to extract other features from the HTTP header to enhance the detection capabilities.

Fig. 1. URL composition structure

C. HTTP Header Fields

There is often a difference between the malicious and the legal traffic on both the request and the response *Header Fields* (HF). In this paper, we mainly inspect two fields:

- Content Type. The HTTP protocol carries data transmission of various types, such as text, pictures, sounds, video, and others. Legal traffic tends to vary significantly. In contrast, most malware choose text-related values such as *text/html; charset=UTF-8*. The value of the content type could help our model filter some legal traffic to some degree.
- User-Agent. Although the user-agent field has many different values and has a long tail effect, it is still an effective indicator of compromised hosts because malware may carry the fake browser-like information or its own unique identification. In this paper, we prefer to use the one-hot-encoding method to represent the popular user agent rather than the blacklist-based method. In addition, the character distributions can also roughly represent the differences.

In addition to the above fields, another important feature is the HTTP return code. There is a discrepancy between the responses from malicious requests and legal requests. According to our statistics, the proportion of 200 in legal traffic is higher than that in the malware traffic. Being influenced by the C&C server offline and the sinkhole defense, malware traffic's distributions on 301, 403 and 404 are significantly higher than those of the normal traffic.

D. HTTP Header Sequences

Apart from the request line including methods, URI and protocol version, the remaining fields in the request header can be placed in any order. The sequences of header fields from applications are different, and malicious requests will deviate slightly from each other as well as legal traffic through artificial settings. As shown in Fig. 2, typical web browser traffic has a fixed set of headers and orders. In contrast, malicious connections always have a fewer number of header fields and different sequences [17]. To some extent, we define this header fields sequence as a fingerprint, which can be used to indicate the source of the traffic. Given the relative stability of this sequence, attackers often consider changing the User-Agent value but rarely change the appearance and order of the header fields. Therefore, we believe this header sequence can be used to distinguish different types of traffic, which contain malware and legitimate traffic.

Different from the previous method of using header chains [7], we adopt both the N-gram technique to describe the special header sequences, which are able to learn intrinsic properties

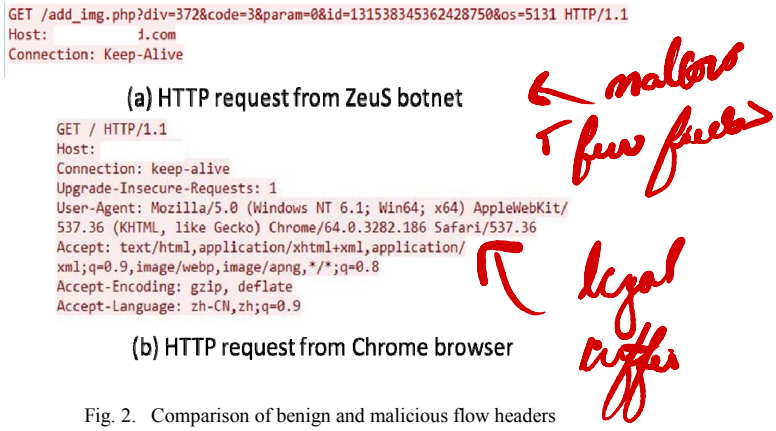

Fig. 2. Comparison of benign and malicious flow headers

and improve generalization capabilities. We could select top features or adopt the hashing trick method to solve the problem of dimension disaster. We believe the sequence feature is complementary to the above feature engineering methods. The combination of these features creates an accurate detection model.

IV. System Design

In this section, we introduce the implementation of our detection approach. The corresponding prototype, called MalHunter, is implemented based on the binary classification model. As shown in Fig. 3, both legitimate and malicious network data (PCAP form) will be sent to the system for training and testing. According to the implementation process, MalHunter can be divided into three core components: a) HTTP feature extraction, b) model training and c) classification. We will introduce the corresponding function of each module.

A. HTTP Feature Extraction

The network traffic from different sources will be parsed and have extraction features in the module. To retain the original information, the input format is generally unified as PCAP. Based on the HTTP flow unit, the module initially filters any other types of traffic and then extracts the HTTP features mentioned in Section 2. Additionally, to ensure the successful implementation of the follow-up progresses, we also conduct necessary feature engineering processing in the training module. For example, we prefer to use L2 regularization for preventing binary classification model training from overfitting to some degree.

B. Model Learning

The feature extracted in the previous step is transformed into the feature vectors, which serve as inputs to the training model. According to our research and public malware analysis reports, it is possible that traffic from malware may contain legitimate traffic as well. For example, some malware will request legitimate websites to test connectivity at the initial stage or visit legitimate websites such as Geoip to get information they are interested in. In our experiment, we use external methods such as Virus Total or other intelligence threats tools, to remove irrelevant traffic whose destinations are normal. Although we may lose some network behaviors from malware, we believe the proportion is relatively small, and the method is able to reduce interference of the noise data.

The core of MalHunter's implementation lies in the classifier based on a machine learning algorithm. Considering the large number of features and the large amount of training data, we choose the random forest and XGBoost as our classification algorithms, which are widely used in the industry and have achieved good results in practical application before. Their great generalization ability satisfies the requirements of similar malicious traffic from the same malware family. We will compare the detection effects of the two algorithms in the next section.

C. Classification

The binary classification model inspects the traffic to decide whether a HTTP connection originated from malware or legitimate web applications. In the training scenarios, we adopt the cross-validation method, which randomly chooses k-1/k dataset to train the model initially and then uses the remaining 1/k datasets for testing. The results reflect the performance of the detection system and support subsequent improvements. In real detection scenarios, after feature extraction, the captured network data from the real world are forwarded to this module, and the classifier judges whether each HTTP flow is malicious or not. It is noted that this paper does not focus on multi-classification to identify families of samples, even though the method is similar. Such discussion is beyond the scope of this paper.

In addition to that mentioned above, we are able to consider introducing extra methods to filter the network data before our detection, which is beneficial to improve the overall detection performance. For example, we can introduce signature-based detection or blacklist and whitelist in the front of our classifier, which can shorten the detection time by removing the black or white traffic from the mass data. In addition, our detection method can easily be combined with other special detection engines, such as the DGA detection system [18], to improve the overall detection capabilities.

V. Experiments

In this section, we describe the strategy for data collection. The Tcpdump tool is used to collect the flow data from sandboxes and volunteer PCs and then save the data as the PCAP format. We also describe the cross-validation and test experiments in detail.

A. Malware Data and Benign Data

The malware samples were collected from September to October 2017 from Virus Total, covering 10 botnet families such as ZeuS, Spyeye, Necurs, Dridex, Lokibot and others. We run these samples in Cuckoo Sandbox for 5 minutes, whose operation system uses Windows XP x86 or Windows 7 x64. Then, the network activities are recorded using Tcpdump. Finally, we obtain a total of 33,374 HTTP flows for training and cross-validation.

Legitimate traffic mainly comes from volunteers in our local LAN. We deploy monitoring scripts in their client devices, recording their access to legitimate websites and application traffic. To avoid possible noisy traffic, we used IDS to filter the suspicious traffic from the user data. As many current sites adopt HTTPS protocols for transmission and our inspections do not check the communication content, there are limited privacy concerns in our experiment. Within seven days, we collected more than 1 million HTTP flow data.

To test the detection ability of our classification model, in January 2018, we deployed our system on real enterprise networks, used the classifier that is trained with the above data to evaluate the detection ability of new network threats.

B. Detection Evaluation

We use 10-fold cross validation for benign and malicious datasets to evaluate the performance of the MalHunter. The malware HTTP flows are assigned to every fold on average, each fold would contain nearly 3337 malicious flows from all 10 malware families. There are more than 112540 benign flows in each fold, which is much larger than the number of malicious traffic. However, in the real world, the value of malicious traffic is often far less than the number of legitimate

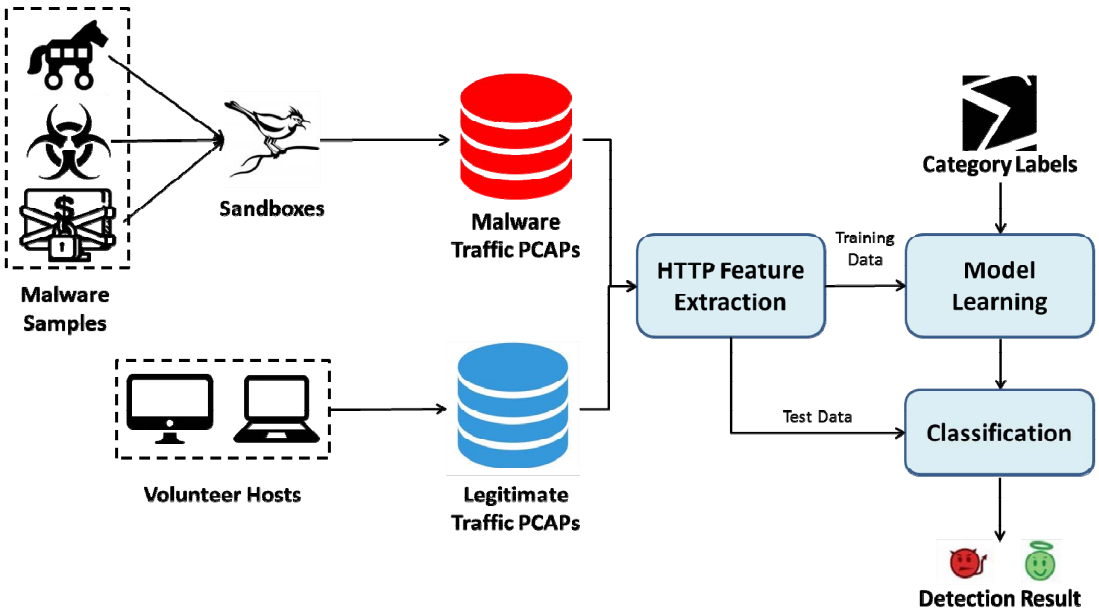

Fig. 3. The overview of MalHunter system

traffic, so we think this ratio is reasonable.

This experiment mainly adopts the following indicators to evaluate the overall performance:

- False positive (FP), the ratio of the number of benign traffic that is wrongly judged as malicious traffic divided by all the benign traffic;
- Recall rate, the ratio of the number of detected malicious traffic divided by the total number of malicious traffic;
- Precision rate, the ratio of the number of correct malicious flows in positive results divided by the number of positive results.

In addition to the above indicators, we usually use the ROC curve or AUC value to evaluate the overall performance of the model.

Table 1 shows the results of the 10-fold cross validation. We set up 200 estimators in both the random forest and XGBoost models. According to the precision and recall rate, we can see that the XGBoost model is better than the random forest model. What's more, the performance of model trained by all three types of features is better than that of not using the *HTTP headers sequence* (HS) features, which proves that the use of HS features can further improve the accuracy of detection(the precision rate reaches 98.32%) malware traffic, and the false alarm rate is still low (0.05%). The AUC values of both classification models are greater than 0.99.

After that, we test whether our detection system can accurately identify malicious traffic from the family samples that are found in the real enterprise network. The binary classification model is trained using the above training data set. In the volunteer network, our model found the 277 malicious traffic in more than 10 million HTTP traffic, we confirms the malicious traffic derived from the new malware variants by threat intelligence from Virus Total. The Fig. 4 shows the top 12 malicious URLs in the detected result, 9 of them our classification model have not seen before. This result proves that the model learned the traffic behavior pattern from the malware samples for discovering the unobserved variants thorough network traffic analysis.

TABLE I. CLASSIFICATION RESULTS OF DIFFERENT MACHINE LEARNING ALGORITHMS

Algorithm	Features	Precision Rate	Recall Rate	False Positive
Random Forest	url+HF	94.65%	98.29%	0.16%
Random Forest	url+HF+HS	95.06%	98.57%	0.15%
XGBoost	url+HF	96.98%	98.64%	0.09%
XGBoost	url+HF+HS	98.32%	98.70%	0.05%

VI. DISCUSSION

Although the previous chapters verify the effectiveness of our method based on real network data, there are still two major problems: evasion attack and HTTPS protocol.

There are two kinds of evasion methods used by malware. First, some malware samples would inject noise traffic that belong to benign websites into their request sequences, which interface with the detection and analysis and cloak their real C&C communications. In this situation, we could adopt a pre-filtering mechanism in the training phase to avoid classification errors to benign websites [19]. Second, the malware may cloak its true malicious traffic, such as by disguising itself as browser traffic, or requesting public third-party Web services for stealthy C&C activities (e.g., Pastbin, Github). From the

Use of some algorithm for with different features *

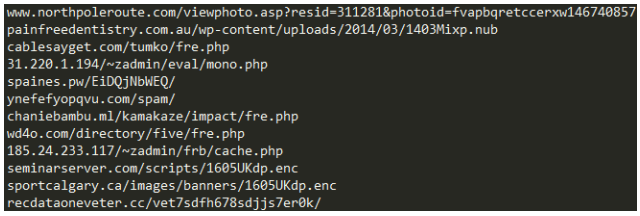

Fig. 4. Malicious URL detected in the real enterprise network.

statistical analysis, these evasion attacks will result in false detection or omission. We believe the threat intelligence approach can be used as a supplement.

Our method cannot handle malicious software that uses HTTPS to encrypt communications because the header content is encrypted. The detection of such malware is another large area of research, requiring more diverse data and more complex processes. We believe changing the sources of feature collection can also be used to detect encrypted traffic, such as the characteristics of the TLS handshake packages. We will perform these related studies in the future.

VII. CONCLUSION

This paper proposes a precise detection method suitable for classifying HTTP-based malware behaviors. It combines the statistical features from different contents such as *character distribution of the URL, HTTP header fields and HTTP header's sequence,* achieves robust detection compare to widely-used signature-based approach. The behavior representation is designed to be invariant under changing C&C server response contents by botmaster, allowing the corresponding classifier is capable of detecting infected hosts accurately.

We made the demo of the proposed method and deployed it on local network for evaluating the detection performance under real traffic, which contain 33,374 malicious HTTP flows covering 10 malware families and more than one millions begin HTTP flows. The 10-fold cross validation experiment result show that the adoption of *header's sequence* features could further improve the classifier's precision rate (98.32%). Moreover, the trained model is able to discover unseen variants before in enterprise network, it is proved that the method can effectively recognize the traffic flow originate from same malware family. We believe the proposed approach is a great complement to the existing signature-based detection methods.

REFERENCES

[1] Perdisci R, Lee W, Feamster N. Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces [A]. // Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation NSDI [C], Berkeley, CA: USENIX Association, 2010:10-14.

[2] Perdisci R, Ariu D, Giacinto G. Scalable fine-grained behavioral clustering of http-based malware[J]. Computer Networks, 2013, 57(2): 487-500.

[3] T. Berners-Lee, R. Fielding, L. Masinter, Uniform Resource Identifier (URI): Generic Syntax, available from: http://www.ietf.org/rfc/rfc3986.txt (2005)

[4] Anderson B, McGrew D. Identifying encrypted malware traffic with contextual flow data[C]//Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security. ACM, 2016: 35-46.

[5] Bartos K, Sofka M, Franc V. Optimized Invariant Representation of Network Traffic for Detecting Unseen Malware Variants[C]//USENIX Security Symposium. 2016: 807-822.

[6] Kirubavathi G, Anitha R. Botnet detection via mining of traffic flow characteristics [J]. Computers & Electrical Engineering, 2016, 50(1): 91-101.

[7] Zarras A, Papadogiannakis A, Gawlik R, et al. Automated generation of models for fast and precise detection of HTTP-based malware [A]. // Proceedings of the 2014 Twelfth Annual International Conference on Privacy, Security and Trust (PST) [C], Piscatawary, NJ: IEEE, 2014: 249-256.

[8] K. Rieck, G. Schwenk, T. Limmer, T. Holz, P. Laskov, Botzilla: Detecting the "phoning home" of malicious software, in: Proceedings of the 2010 ACM Symposium on Applied Computing, SAC '10, ACM, New York, NY, USA, 2010, pp. 1978–1984.

[9] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, C. Kruegel, Exposure: A passive dns analysis service to detect and report malicious domains, ACM Trans. Inf. Syst. Secur. 16 (4) (2014) 14:1–14:28.

[10] Sharifnya R, Abadi M. A novel reputation system to detect DGA-based botnets [A]. // Proceedings of the 3th International Conference on Computer and Knowledge Engineering (ICCKE 2013) [C], Piscatawary, NJ: IEEE, 2013: 417-423.

[11] Schiavoni S, Maggi F, Cavallaro L, et al. Phoenix: DGA-based botnet tracking and intelligence [A]. // Proceedings of the 2014 International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment [C], Berlin: Springer, 2014: 192-211.

[12] Gu G, Perdisci R, Zhang J, et al. BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection[C]//USENIX security symposium. 2008, 5(2): 139-154.

[13] Lu C, Brooks R. Botnet traffic detection using hidden markov models [A]. // Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research [C], New York: ACM Press, 2011.

[14] Grill M, Rehák M. Malware detection using HTTP user-agent discrepancy identification [A]. // Proceedings of the 2014 IEEE International Workshop on Information Forensics and Security (WIFS) [C], Piscatawary, NJ: IEEE, 2014: 221-226.

[15] Saxe J, Berlin K. eXpose: A Character-Level Convolutional Neural Network with Embeddings For Detecting Malicious URLs, File Paths and Registry Keys [J]. arXiv preprint arXiv:1702.08568, 2017.

[16] Machlica L, Bartos K, Sofka M. Learning detectors of malicious web requests for intrusion detection in network traffic [J]. arXiv preprint arXiv:1702.02530, 2017.

[17] Hao Dong, Chenhuai Lu, David Yu, et al. BEYOND THE BLACKLISTS: DETECTING MALICIOUS URL THROUGH MACHINE LEARNING. https://www.blackhat.com/docs/asia-17/materials/asia-17-Dong-Beyond-The-Blacklists-Detecting-Malicious-URL-Through-Machine-Learning.pdf.

[18] Woodbridge J, Anderson H S, Ahuja A, et al. Predicting domain generation algorithms with long short-term memory networks [J]. arXiv preprint arXiv:1611.00791, 2016.

[19] Denis Sinegubko. GitHub Hosts Infostealer. https://blog.sucuri.net/2018/03/github-hosts-lokibot-infostealer.html