ISSN 2319-2720
Volume 7, No.2, April – June 2018
International Journal of Computing, Communications and Networking
Available Online at http://www.warse.org/ijccn/static/pdf/file/ijccn15722018.pdf
https://doi.org/10.30534/ijccn/2018/15722018

Detecting malware infected devices by discriminating legitimate from malicious traffic using HTTP protocol

Aslam Parvej[1], Bibesh Poudel[2], Deepak P[3],Vinod Kumar[4] , Prof. Madhura G Sunil[5]
[1]Student of EWIT, India,Imaslamparvej@gmail.com
[2]Student of EWIT, India, bip.terence@gmail.com
[3]Student of EWIT, India, deepakjvr619@gmail.com
[4]Student of EWIT, India, vinukumar0911@gmail.com
[5]Professor at EWIT, India, madhurags@gmail.com

ABSTRACT

Malware causes a huge damage is one of the serious problems that has to be identified. The evasion malware spread in recent times made it difficult to detect in pre-infection time. The best approach is malware detection at post-infection timing. By monitoring the internet traffic this work aims to identify the malware-infected devices. Most of the malware uses the internet as a means to communicate with the command and control servers which is located on the external network, the recorded HTTP headers information is monitored to discriminate between the legitimate and malicious traffic. This method is scalable and robust because it uses automatic template generation which will reduce most amount of information that has to be kept while obtaining high accuracy in classification. We us several classifiers, which makes use of extracted templates and classifies traffic into two categories: malicious and legitimate.

Keywords: Classification; malware datasets; automatic template generation; network based detection.

1. INTRODUCTION

As there are many types of malware, all operating systems and devices are affected by malware. Any devices could be infected with new malware, which cannot be detected with existing solutions that is malware detection at pre-infection is not possible to some extent. Recent increase in spread of evasion malware has made it difficult to detect malware at the pre-infection timing. So detecting the malware at post-infection timing is a promising approach that can fulfill this gap. The main aim is to detect the malware infected devices as early as possible.

Malware infected devices use to communicate with external command and control (C&C) servers, we make a supposition that Internet traffic is useful approach to detect malware infected devices. The main advantage of using the traffic measurement based approach is that it allows to monitor large numbers of clients. If we find any client as a source of malware/suspicious traffic and based on threshold that client is likely a malware-infected device. The majority of malware uses HTTP protocol as to communicate with command and control servers which reside in the external network. We classify traffic into two categories: malicious and legitimate. To reduce the amount of information to be kept while achieving the high accuracy of classification, we use automatic template generation. This automation enables us to make our system robust to change in the malware traffic. This can detect malware infected devices accurately in a scalable manner.

2. RELATED WORKS

In this section, we review several studies that used HTTP information for detecting malware [15], [17], [21], [23]. We also discuss how our approach is different from these past studies.

Zhang et al. [23] made use of the User-Agent field as the useful feature for detecting malware. They demonstrated that regular

expression could be used to characterize HTTP headers. They also confirmed that a fake User-Agent could be identified together with the information extracted with the OS fingerprinting technique. Grill et al. [17] also used the User- Agent field to detect malware communication. They found that User-Agent can be classified into the five patterns: Legitimate user s browser, Empty, Specific, Spoofed, and Discrepant. According to their findings, some malware uses User-Agent of the web browser used by the actual owner of the device; in such a case, it's hard to detect malware communication simply from the User-Agent information. Nelms et al. [21] proposed a system called ExecScent, which is the closest work to ours. It aims to detect bot using the entire HTTP header fields.

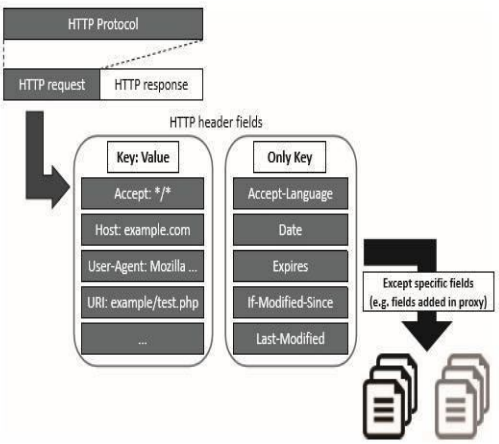

Fig 1: The extraction of HTTP header.

They manually created templates using the domain knowledge, i.e., URL path, query, User-Agent, etc. They characterize the templates with a regular expression. We note that our approach automated the template generation scheme; thus, we do not need to employ manual inspection to create the useful templates. Chiba et al. [15] developed a system called BotProfiler. BotProfiler is a system that aimed to improve the performance of ExecScent by using URL path, URL query, and User-Agent information. Like the ExecScent system, the BotProfiler system also requires building manually crafted templates. While all these past studies heavily rely on the domain knowledge when they extract useful features to detect malware, our approach aimed to automate the feature extraction IEEE ICC 2017 Communication and Information Systems Security Symposium process by making use of the automatic template generation algorithm. We also note that DNN enables us to perform the feature learning; i.e., it automatically expresses useful features with the neural network.

We believe that our approach has an advantage over the strategies used in the past studies because ours is robust to the change in the features of malware communication.

3. METHODLOGY

Step 1: Extraction of header

The malware uses HTTP protocol to communicate with the main command and control server, the http header is then extracted with a specific keys and values and it uses the URI which is related to HTTP is extracted for keys, values and information from URI, we use the "URI" as key source of information. The unwanted keys and values are eliminated depending on the data collection environment from the extracted header such as date, last modified status, expires. The http header are eliminated if appearing 10 times in training data and extra fields are added when packets are passed through proxy servers and it passes through middle boxes which are not used for data inclusion.

Step 2: Automatic template generation

The http header contains n number of fields which is tough to identify and we aim in constituting the header fields and compress their information. We make use of template generation technique such as DBSCAN.

1) *Scoring*: We use the http header field in training data, we find values or scores to each field and each words within the field .each field is divided by a set of separators and we check the probability of each word as the score, for a word x, in a given field Y, the score of the word, S(x,Y), is given by the following probability

$$S(x)=P(x|pos(x,Y),len(Y))=n(x,pos(x,Y),len(Y))/n(\ pos(x,Y),l\ n(Y))$$

Where pos(x, Y) is the position of the word in the given field, Y, and len(Y) is the number of words in the field, Y, respectively. If Y = {foo, bar, baz,

qux} and x= bar, pos(x,Y) = 2 and len(y) = 4. n(X) is the number of occurrences of the variate X over the entire fields.

2) *DBSCAN*: DBSCAN is one of the clustering algorithms that do not require a predefined number of clusters and its algorithm extracts clusters with any shape. The thresholds for the minimum distance and the minimum number of elements in a cluster are given by the DBSCAN algorithm.

3) *Clustering*: Clustering is used to combine the scoring method and the DBSCAN results. The threshold of the score and DBSCAN threshold be minimum, the two thresholds δ (δ ≥ 0) and β (0 <β<1) it will be determined later. The δ is the threshold that determines the minimum distance between clusters sorting the words. The words are sorted on score value in descending order. Cluster. DBSCAN algorithm is used to cluster, when the value of the score is less than 0, the next word is added to current cluster. The method is repeated until all the words are included in the cluster. Output template generation. The cluster results is obtained by template generation, which are clustered into a group using the templates.

Step3: Tra□c classification using the system template generation

We detect the malicious traffic using classifiers, the classification of http is done through classifiers, random forest and support vector machine (svm) are used for the classification of templates because they give the best performance, the trained datasets are classified and compared with the packets from the trained.

The most suitable values for each parameter of SVM and RF were determined by 5-fold cross validations. The DNN model consists of four layers: an input layer, two hidden layers, and an output layer. The optimization is performed by Adaptive Moment Estimation. The error and activated functions are based on cross-entropy and Rectified Linear Unit, respectively.

Datasets

1. Training datasets:

a. Malicious datasets:

The training data for malicious traffic detection were compiled from few sets of their traffic data. Each datasets are trained with a set of malware samples. These were collected using the server-type and client-type honeypot systems and were analysed with a commercial dynamic malware analysis system that is connected to the Internet. The system uses these internet to collect the malware samples and these are placed into a trained dataset.

b. Legitimate traffic:

As a training dataset of legitimate traffic is detected using the classifier. The URL registered with Malware Domain Block list will detect the malware affected datasets and block the dataset, the program of the dataset is small so the classifier will ignore the dataset and term the dataset as legitimate traffic.

2. Test data:

a. malicious tra□c:

The test data of malicious tra□c was compiled from some samples and their tra□c data. To avoid bias among samples in the same resource, we collected samples. These samples were judged malicious by at least one anti-virus vendor in VirusTotal. No overlaps existed between the training and test data. The tra□c data of the malware samples were collected over the same period as the malware collection using Cuckoo Sandbox that is dynamic malware analysis system of open source software. It is also connected to the Internet and analyses on Windows machine to execute malware samples for 1.5 min.

b. legitimate tra□c:

The network observation data were acquired from the same organization as the test data of the legitimate traffic in September 2016. We also disposed of any packets resembling malicious packets.

4. EXPERIMENTAL RESULT

In this paper, we present how we determined the threshold used for the automatic template generation. We then show the primary results, the accuracy of the classification models. Finally, we demonstrate the effectiveness of the automatic template generation.

A. Automatic template generation

We first present how we set the threshold β used in the automatic template generation. shows how the number of templates depends on the threshold,

β. δ = 100 corresponds to the case where no templates are created. The smaller the δ and β, the more e□ective the template creation.

The decrease in the number of templates is especially remarkable at β = 0.5. Figure 5 shows the CDFs of len (F). As shown in the graph, roughly 50% of the fields had len(F) = 2. This result indicates that templates of these fields are

created when β is less than 0.5. Given these observations, we set β = 0.5 as the most suitable threshold.

B. Effectiveness of Automatic template generation

We study how the automatic template generation algorithm a□ects our results. For all the templates extracted with δ = 10−1 and β = 0.5, we calculated the mutual information (MI), which is one of the metrics that can tell you the contribution of a given feature to the classification. MI is formulated as follows:

$$MI(X;Y) = \sum_{i=1}^{m}\sum_{j=1}^{n} p(x_i, y_j)\log_2 \frac{p(x_i, y_j)}{p(x_i)p(y_j)},$$

where p(xi) refers to probability of xi in random variable X = {x1,x2,...,xm}, p(yj) refers to probability of yj in random variable Y = {y1,y2,...,yn}, and p(xi,yj) expresses the simultaneous occurrence probability of xi and yj. Table V shows the top-10 templates that had the largest MI. As we see from the table, the templates with the wildcard characters inserted had high MI. This result indicates that automatic template generation was e□ective not only in reducing the number of templates but in extracting useful features that contributed to the classification.

5. CONCLUSION

We proposed a system called proxy server for detecting malicious traffic, thereby searching for malware-infected devices. The key ideas of our research were to create templates automatically for gathering information of each HTTP header field and using support vector machine technique for detection. As a result of the extensive experiments using large-scale datasets, we demonstrated that our trained system successfully detected malicious traffic with up to 97.1% precision and a low false detection rate below 1.0%. One key technical contribution of this work was that introduction of the automatic template generation algorithm, which contributed not only to reducing the amount of information to be kept but also to extracting useful features. We believe that the key ideas and approaches used in this paper are useful for other studies that attempt to classify malicious activities given a large number of features.

REFERENCES

[1] Content security software - Internet Security & Cloud-TrendMicroUSA.http://www.trendmicro.co.

[2] Cuckoo Sandbox: Automated Malware Analysis.https://www.cuckoosandbox.org/.

[3] DNS-BH - Malware Domain Blocklist. http://www.malwaredomains.com/.

[4] ESET predictions and trends for cybercrime in 2016. http://www.welivesecurity.com/2015/12/23/ eset-predictions-for-cybercrime-trends-in-2016/.

[5] Kaspersky Lab |Antivirus Protection & Internet Security Software. http: //www.kaspersky.com/.

[6] Large CCTV Botnet Leveraged in DDoS Attacks. https://blog.sucuri. net/2016/06/large-cctv-botnet-leveraged-ddos-attacks.html.

[7] LIBSVM– A Library for Support Vector Machines.http://www.csie.ntu.edu.tw/∼cjlin/libsv/.

[8] MalShare. http://malshare.com/.

[9] Malwr - Malware Analysis by Cuckoo Sandbox. https://malwr.com/.

[10] Quarterly Threat Report: What Do the Numbers Mean to Me? https: // blogs.mcafee.com/ consumer/ quarterly-threat-report-numbers-mean/.

[11] scikit-learn: machine learning in Python. http://scikit-learn.org/stable/.

[12] TensorFlow - an Open Source Software Library for Machine intelligence. https: //www.tensorflow.org/.

[13] VirusShare.com. https://virusshare.com/.

[14] VirusTotal - Free Online Virus, Malware and URL Scanner. https://www.virustotal.com/.

[15] Daiki Chiba, Takeshi Yagi, Mitsuaki Akiyama, Kazufumi Aoki, Takeo Hariu, and Shigeki Goto. Botprofiler: Profiling variability of substrings in http requests to detect malware-infected hosts. In Trustcom/ BigDataSE/ISPA, 2015 IEEE, volume 1, pages 758–765. IEEE,

[16] Martin Ester, Hans-Peter Kriegel, J¨org Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In Kdd, volume 96, pages 226–231, 1996.

[17] Martin Grill and Martin Rehak. Malware detection using http user agent discrepancy identification. In Parallel Computing Technologies (PARCOMPTECH), 2015 National Conference on, pages 221–226. IEEE, 2015.

[18] John P John, Alexander Moshchuk, Steven D Gribble, and Arvind Krishnamurthy. Studying spamming botnets using botlab. In NSDI, volume 9, pages 291–306, 2009.

[19] Tatsuaki Kimura, Keisuke Ishibashi, Tatsuya Mori, Hiroshi Sawada, Tsuyoshi Toyono, Ken Nishimatsu, Akio Watanabe, Akihiro Shimoda, and Kohei Shiomoto. Spatio-temporal factorization of log data for understanding network events. In IEEE INFOCOM 2014-IEEE Conference on Computer Communications, pages 610–618. IEEE, 2014. https://doi.org/10.1109/INFOCOM.2014.6847986

[20] Dhilung Kirat, Giovanni Vigna, and Christopher Kruegel. Barecloud: Bare-metal analysis-based evasive malware detection. In Proceedings of the 23rd USENIX Conference on Security Symposium, USENIX Security '14, pages 287–301, Aug. 2014.

[21] Terry Nelms, Roberto Perdisci, and Mustaque Ahamad. Execscent: Mining for new c&c domains in live networks with adaptive control protocol templates. In Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13), pages 589–604, 2013.

[22] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1):1929–1958, 2014.

[23] Yang Zhang, Hesham Mekky, Zhi-Li Zhang, Ruben Torres, Sung-Ju Lee, Alok Tongaonkar, and Marco Mellia. Detecting malicious activities withuser-agent-based profiles. International Journal of Network Management, 25(5):306–319, 2015 https://doi.org/10.1002/nem.1900

AUTHORS BIOGRAPHY

Aslam Parvej: pursuing B.E in CSE, EWIT (VTU), Bengaluru. His areas of interest are Computer Security, Databases, Computer Networks, Storage Area Networks, Programming the Web, Software Engineering, Cloud Computing, Computer Graphics, etc.

Bibesh Poudel: pursuing B.E in CSE, EWIT (VTU), Bengaluru. His areas of interest are Computer Security, Databases, Computer Networks, Software Engineering, Java & Python Programming etc.

Deepak P: pursuing B.E in CSE, EWIT (VTU), Bengaluru. His areas of interest are Computer Security, Databases, Computer Networks, Computer Graphics, Software Engineering, Internet of Things etc.

Vinod Kumar S: pursuing B.E in CSE, EWIT (VTU), Bengaluru. His areas of interest are Computer Security, Databases, Computer Networks, Storage Area Networks, Programming the Web, Software Engineering, Cloud Computing, Computer Graphics, etc.

Madhura G Sunil: Professor in Computer Science & Engineering, East West Institute of Technology (VTU), Bengaluru. Qualification: B.E, M.Tech,. Her areas of research are Wireless Ad-hoc Networks, Computer Networks, Software Engineering, Genetic Algorithms, Machine Learning and Cloud Computing.