A Research on Clustering and Identifying Automated Communication in the HTTP Environment

Manh Cong Tran, Nguyen Quang Thi, Nguyen The Tien, Nguyen Xuan Phuc and Nguyen Hieu Minh

Abstract A lot of HTTP traffics are unnoticed to users because they are automatically generated from software. This caused by HTTP protocol characteristics. For the purpose of communication with servers, HTTP-based applications always automatically and actively send requests to their hosts because HTTPs are designed as connectionless protocols. In addition, all kinds of HTTP communications from software such as a bot, adware, and normal web accesses are mixed clearly. This raises the requirement for clarification of HTTP traffics. Most previous studies concentrated on HTTP-based malicious bot traffics, however, graywares such as adware or unauthorized applications are also becoming serious internal threats since they can stealth sensitive information or web usage experiences from infected systems. In this study, a new method for clustering and identifying HTTP communications is proposed. It focuses on analyzing of HTTP-based software Internet access behaviors. The method is tested with real outbound HTTP communication of a private network. Examination showed improved results with an accuracy rate of 91.18% in clustering and identifying HTTP automated communications.

Keywords HTTP traffic analysis · Malware · Grayware · Suspicious · Botnet · Clustering · Identification · Detection

M. C. Tran (✉) · N. Q. Thi · N. T. Tien · N. X. Phuc
Le Quy Don Technical University, Hanoi, Vietnam
e-mail: manhtc@gmail.com

N. Q. Thi
e-mail: thinq.mta@gmail.com

N. T. Tien
e-mail: tiennt@mta.edu.vn

N. X. Phuc
e-mail: phuc.nx89@gmail.com

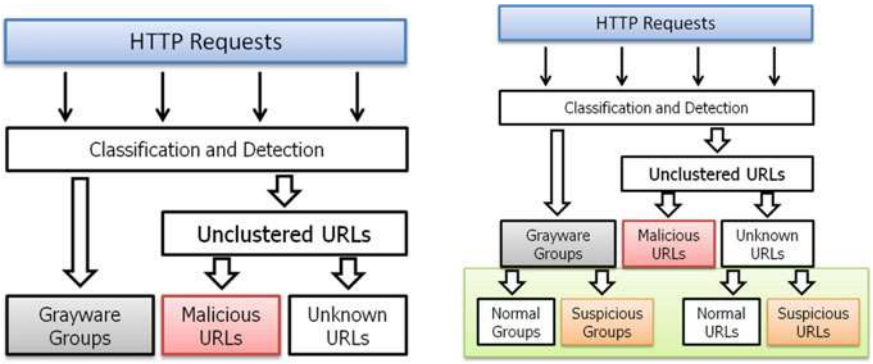
N. H. Minh
Academy of Cryptography Techniques, Hanoi, Vietnam
e-mail: hieuminhmta@gmail.com

V. K. Solanki et al. (eds.), *Intelligent Computing in Engineering*,
Advances in Intelligent Systems and Computing 1125,
https://doi.org/10.1007/978-981-15-2780-7_111

1 Introduction

Automated communications in HTTP environment would not only be generated from normal applications such as antivirus updaters but could be generated from grayware like adware, unauthorized or malicious HTTP software. Malicious requests' structure is similar as that of legitimate normal requests and their traffic mixes adequately with each other. Therefore, malicious and normal activities distinction from HTTP traffic is really a big challenge in HTTP communication environment especially large traffic are created every day. Being different from direct TCP/IP connections which are connection orientated, HTTP-based communication is connectionless protocol. So that to retain the updates or to receive commands from hosts, HTTP-based software follows pull method, where they actively send requests to their servers. However, particularly, there are complex contrasts in the communications behavior of various kinds of HTTP-based applications to their sites.

HTTP malware or C&C channel detection has focused in most of the previous researches such as [1, 2]. However, internal threats within/to a network are also increasing from other suspicious software such as adware or spyware with the intention of HTTP communication behavior analysis, access graph, which is extracted from request based features, is presented in [3]. Based on that, a method to classify and also to detect HTTP automated traffic is proposed as can be seen in Fig. 1a. Accordingly, HTTP automated traffics are clustered into groups based on their behavior and detected as grayware traffic (unknown and unnoticed traffic). With unclustered ULRs, they will be detected as malicious through a score. However, in grayware group, it is possible to consider to identify more details that they are either suspicious or normal group. In addition, unknown traffic in [3] still needs to be clarified. To overtake that sufficient, currently, the behaviors of each type of HTTP-based software are analyzed in more detail with additional new key features. Based on that, a new method is proposed in clustering and identifying a type of communication.



(a) Main results in previous work in [3] (b) Addition flow in this work

Fig. 1 Research targets

Accordingly, HTTP automated traffics are not just classified but also identified into three kinds of traffics, normal, suspicious and malicious, based on HTTP access behaviors as summarized in Fig. 1b.

2 Related Work

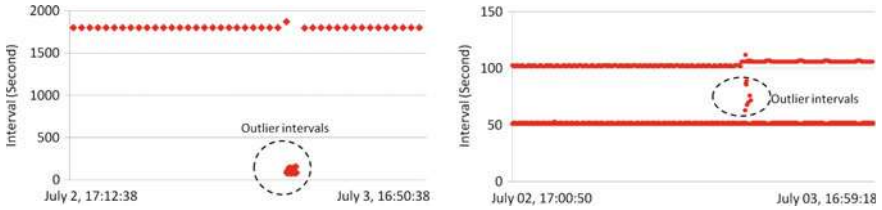
Regular defense mechanisms such as in Antivirus (AV) products are the most common content-based malware detection techniques. However, some works [4, 5] discover that primary AV engines just detect only 30–70% of recent malwares by using signature-based techniques.

Defeating the issue of content-based detection researches, many studies indicate to use network traffic analysis approaches [1, 2]. In CoCoSpot of [1], they proposed a clustering method to analyze relationships between botnet C&C flows and an approach to recognize botnet command and control channels solely based on traffic analysis features. To achieve this, the authors collected different parameters of the network traffic and consider to response message length from the server. However, in many C&C servers, the length of each response message is not stable or even there is no response in each request. Thus, the detection result might decrease in those cases. In [2], a method which applied discrete time series is analyzed to examine the aggregated traffic behavior in order to detect botnet C&C communication channels traffic. This research focus on the detection of botnet traffics to C&C servers, but it is confirmed that HTTP-based threats also can be from other types of automated software such as HTTP spyware, adware, or unauthorized applications. Our method tried to cluster and identify HTTP communication by their purposes, not just only for C&C channel, and all selected features are extracted from core properties of HTTP traffic.

3 Methodology and Proposed Method

3.1 Access Graph

Access graph (AG) is presented in [3] which duration of requests to a URL of a clients IP is used. This graph presents the communication behavior of a client to a URL in a specific period as shown in Fig. 2. Assuming that $R = (r_1, r_2, \ldots, r_N)$ is a set of requests from a client to a server/webpage, and all r_i have the same webpage/server URL. In an AG the X axis is the timing of a request and the Y axis presents the request interval value in seconds. Each HTTP application in a client will generate a different AG for each URL and can present the behavior in communication between software and the webpage or server URLs.



(a) An AG from a bot to C&C server (b) An AG from a bot to C&C server

Fig. 2 Almost no variation in AG of two malicious bots to their C&C servers

3.2 HTTP Access Behavior Analysis

AG can present the behavior in communication between software and the webpage or server URLs. Access behavior analysis of programs using AG similarity is presented detail in [3]. In this paper, by detailed observation, some more characteristics are recognized. Suspicious software, for instance—adware, since they update contents, like other HTTP used tools, accesses to many URLs. However, they will collect data from many URLs of multiple sites which own various domain names. This is not alike with normal software, such as an electronic newspaper, since it self-refreshes the contents of presenting page by accessing to many URLs but with only one domain name. A suspicious software starts with the time of human-computer start. Therefore, it is expected that the access duration of a suspicious one might be similar to users' computer interaction. With the intention of the similarity measurement between any AG, a graph distance is proposed in the next Sect. 3.3.

3.3 Access Graph Similarity

It is assumed that there are two access graphs: A and B, $A = (a_1, a_2, \ldots, a_N)$, $B = (b_1, b_2, \ldots, b_M)$. With aim of doing the similarity evaluation between any two access graph A and B, a distance is proposed. It is founded on the Modified Hausdorff (MH) distance which is presented in [6].

First, Euclidean distance $d(a_i, b_j) = ||a_i - b_j||$ is defined as distance between two points a_i and b_j.

Second, $d(a_i, B) = min||a_i - b_j||$ where $b_j \in B$ is determined the distance between point a_i and graph B. Generalized Hausdorff distance [6] for A and B is defined as

$$d(A, B) = \frac{1}{N} \sum_{a_i \in A} d(a_i, B) \tag{1}$$

Finally, the modified Hausdorff (MH) distance [6] between graph A and B is

$$S(A, B) = max(d(A, B), d(B, A)) \tag{2}$$

The smaller the MH distance between graphs A and B, the more graphs A and B are similar to each other. Therefore, MH is also similarity score of A and B. With the target of supplementary for clustering stage, two types of distances which are based on MH in Eq. (2) are proposed. They are max and average distance between any access graph of a group URLs, and are defined as below:

$$MaxS(G) = max(S(U_i, U_j)_{\forall U_i, U_j \in G}) \tag{3}$$

$$AvgS(G) = average(S(U_i, U_j))_{\forall U_i, U_j \in G}) \tag{4}$$

In Eqs. (3) and (4), G is group of URLs, and U_i and U_j are any access graphs of URLs in G.

4 Preprocessing Stage

All stages of prosed method are presented in Fig. 3, in this section, preprocessing stage is provided. This stage is objective to exclude useless processed data or HTTP communication is beneficial for users and is previously presented in [3]. According to that, three procedures are employed: First, a white list of second-level domain

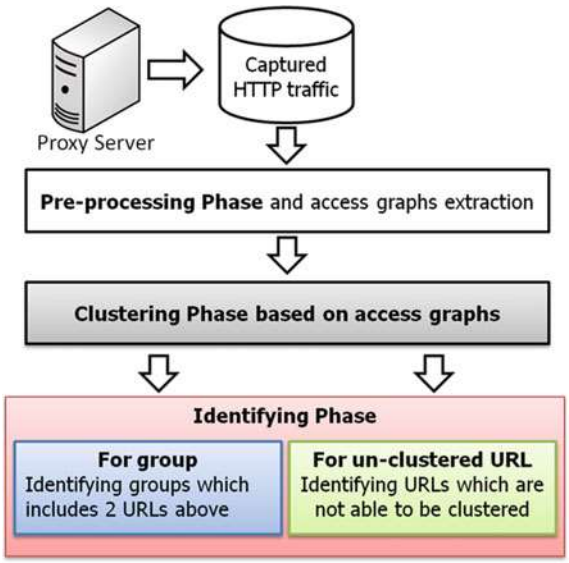


Fig. 3 Main flows and stages of proposed method

name (SLDN) is applied to filter URLs requests from Client IP. Second, the number of requests to a URL from a client IP address is used. It can be seen that suspicious program approach several times to a URL in a duration of time. Therefore, it might not to request by an automated software if the requests number to an URL is too small. The third, URLs which are requested with exceptionally fast in a time duration will represent a malicious application traffic.

5 Clustering Stage

In clustering stage, remaining URLs will be clustered into various groups by using their features which are described in Sect. 3.2. Previous proposed clustering stage is suggested in [3]. Accordingly, a seed vector S is chosen constantly, and d_i and d_j, respectively, are modified Hausdorff (MH) distances [6] between the access graph of URL_i and the access graph of URL_j to the S vector. URL_i and URL_j are placed in the same group if $|d_i - d_j|$ is not greater than a threshold.

In this paper, in order to improve the clustering stage, a new algorithm in Table 1 is proposed. Accordingly, there are three main steps by using the characteristics of access time and access graph similarity which are discussed in Sect. 3.2. The first step is to determine group of URLs based on their access time, two URLs are marked as in the same group if they have approximately equivalent access time. The second step will collect the group from result of Step 1 and calculate a threshold δ which is average similarity of any two in URLs. This δ will help cluster remaining URLs based on their access graph similarity. The last step continuously cluster remaining URLs after Step 1 by checking the similarity of their access graph. In that, two adaptive thresholds instead of using a fixed threshold in [3] are suggested to determine an unclustered URL to be a member of clustered group which is determined in Step 1 or to be paired with other unclustered URL to become a new group.

Table 1 Steps of clustering algorithm

Step	Description
1	For each pair (u_i, u_j) in set U of unique URLs $(1 - M)$. Denoted that (i_{Start}, i_{End}) and (j_{Start}, j_{End}) are timing of start and end request of ui and uj respectively
	If $(i_{Start} \cong j_{Start})$ and $(i_{End} \cong j_{End})$ then u_i and u_j, are in the same group
2	After the first step, part of URLs are clustered, each group owns at-least two URLs. A threshold $\delta = AvgS(U)$ as in Eq. (4)
3	For each URL u_i which is not set group in previous step 1, find a u_j which has minimum MH distance to u_i denote the distance between u_i and u_j is $minS_i$
	u_i and u_j are in the same class if one of two bellow conditions is matched: – if $minS_i < \delta$ in the case u_j is still not set to any group yet – if $minS_i < MaxS$(Group of u_j) (as Eq. (3)) in the case u_j was already set to a group

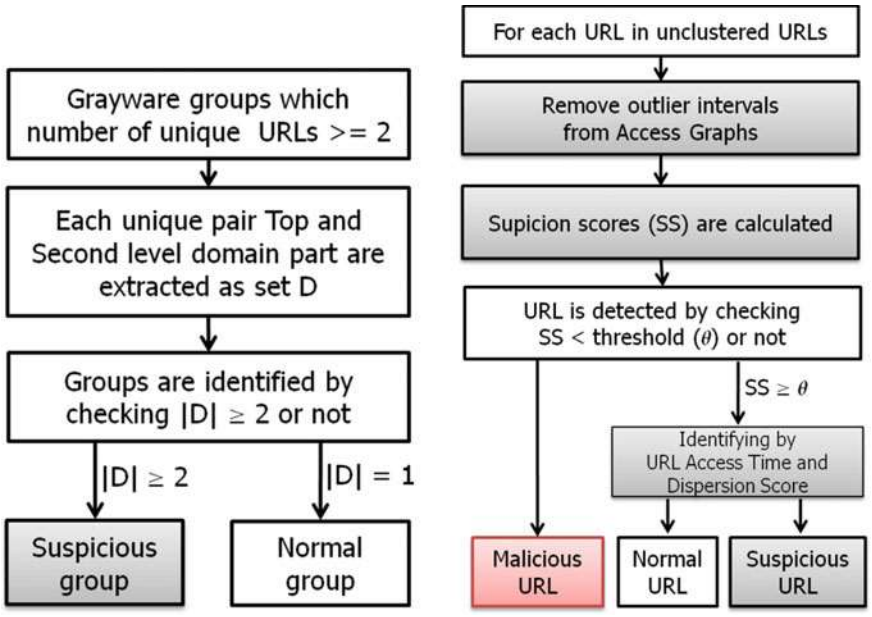
6 Identifying Stage

6.1 Group Identifying Stage

In this stage, groups of two URLs above will be identified into two types of normal or adware groups. The steps for this stage are shown in Fig. 4a.

6.2 URL Identification Stage

For unclustered URLs, a suspicion score is presented to recognize a URL is malicious or normal. Malicious bot always communicate to a specific URL or resource by automatically generating requests with a stable interval (Sect. 3.2), therefore, malicious bot access graph almost has no variation. In this stage, the method does not try to detect the interval of malicious requests but in target to score the variation of access graph. However, the interval of malicious bot requests is changed some times, these are outlier intervals, but the main interval is steady. As can be seen in Fig. 1a, some intervals are uncertain but stable interval is around 1800 s. The number of points in access graph outlier intervals in access graph is minor in comparison



(a) Group identifying stage (b) Unclustered URL identifying stage

Fig. 4 Identifying stage

with main stable intervals for the purpose of outlier intervals detection from AG X, a density-based algorithm (DBSCAN) in [7] is recommended. DBSCAN will help cluster all the similarity intervals in access graph X, from there, outlier intervals are clustered in group with smaller number of members. Details of DBSCAN algorithm is in [7].

The flow for this stage with three main steps are shown in Fig. 4b: First, intervals in access graph of URL are clustered into groups by DBSCAN algorithm. Because malicious bot will communicate to their server by main intervals, so groups containing main intervals will own much more members than other groups. Outlier intervals account for the few points in access graph, and they belong to groups in which they contain smallest number of members, for that matter. These groups will be removed from access graph. As can be seen in Fig. 2a, intervals in the dashed circles will be detected by DBSCAN algorithm and remove from access graph before to be processed in next step. Next, in order to identify a URL is malicious or not, a suspicion score is calculated based on its access graph (outlier intervals are omitted). This score is described in Sect. 6.2. If suspicion score of a URL is less than a threshold, this URL will be recognized as being accessed by malicious communication. Remaining URLs are identified by checking the access time and dispersion score. Suspicious software working along with human computer, therefore, if access time to a URL is similar with user computer interaction, URL will marked as accessing from suspicious. Different from malicious bot, communication which is generated from suspicious program such as adware, is alway with variation intervals. Therefore, URL own high-dispersion score (above 0.5 in this experiment) in access graph is also marked as suspicious one. Dispersion score is described in [7].

Suspicion Score After removing the outlier intervals by DBSCAN algorithm. In order to determine the variation of an AG, a Suspicion Score is proposed, from which it shows suspicion of traffics from a client. Assuming that the access graph after removing outlier intervals of a URL U is specified and denoted as $X = (x_1, \ldots, x_N)$. A suspicion score will be defined as *coefficient of variation* of X_{Avg} as follow equation.

$$SuspiciousScore(X_{Avg}) = \frac{\sigma}{\mu} \tag{5}$$

In that σ and μ are standard deviation and mean of X_{Avg} respectively. The smaller suspicious score shows that URL is more suspicious.

Dispersion Score Dispersion score of a URL is to determine the fragment degree of intervals in its access graph. The score is determined by proportion between number clusters of access graph by DBSCAN algorithm and number of requests to a URL. Assuming that N is number of requests to a URL from a client and C is number of groups which are clustered by DBSCAN. The dispersion score is determined as below.

$$DispersionScore(X) = \frac{C}{N} \tag{6}$$

7 Experimental Result and Discussion

In this part, outbound HTTP traffics from are captured in separated files. 70 HTTP access logs data are selected from 430 GB real traffic which is imported to a big database system. Table 2 summarizes test data. Being different with experimental data in [3], which each log data is captured in just one day, new test data are collected from various range of time which not just in one day. As in Table 2, log data is from 150 min to 5 days, after preprocessing stage 58.85% of URLs need to be clustered and identified in next stages. All results are manually checked in VirusTotal online system [8] and McAfee Web Gateway [9].

Remaining 10,942 unique URLs from preprocessing stage, are as input of clustering stage which results are summarized in Table 3. In that, 92.30% URLs (10,100 URLs) are clustered in group. These results indicate that mostly HTTP communicates to software servers with the similar behavior since just about 7.70% URLs are unclustered which are requests with specified behavior. As analysis in Sect. 3.2, these unclustered URLs are considered as malicious communication.

As in Table 3, 1,591 groups of 10,100 clustered URLs are as input data for group identifying stage which is described in Fig. 4a. The identified results are details in Table 4. There are two kinds of groups are detected, normal and suspicious, and evaluated. Normal groups mostly include all URLs which are requested for news or analytic updates. Vice versa, suspicious groups contain URLs which accessed for

Table 2 Experimental data statistic

No.	Item	Values
1	Number of log data	70
2	Total number of requests	16,211,257
3	Max requests in a log data	3,030,216
4	Min requests in a log data	2,110
5	Max access time in a log data	150 min
6	Min access time in a log data	5 days
7	Requests after preprocessing stage. (58.85% of total requests remaining)	9,540,608 (58.85%)
8	Number of Unique URLs after preprocessing stage	10,942

Table 3 Clustering stage results

No.	Item		Values	Percent (%)
1	Unique URLs after preprocessing		10,942	100
2	Clustered	Groups ($\|$ URLs$\| \geq 2$)	1,591	
		Number of URLs	10,100	92.30
3	Unclustred	Number of URLs	842	7.70

Table 4 Group identifying results

No.	Group type	Groups	Detect results		True		False	
			URLs	Percent (%)	URLs	Percent (%)	URLs	Percent (%)
1	Normal	627	2,476	24.51	2,325	93.90	151	6.10
2	Suspicious	964	7,624	75.49	7,021	92.09	603	7.91
3	Total	1,591	10,100	100	9,346	92.53	754	7.47

Table 5 URL identifying results

No.	URL type	Detect results		True		False	
		URLs	Percent (%)	URLs	Percent (%)	URLs	Percent (%)
1	Normal	399	47.39	316	79.20	83	20.80
2	Suspicious	389	46.20	264	67.87	125	32.13
3	Malicious	54	6.41	51	94.44	3	5.56
4	Total	842	100	631	74.94	211	25.06

Table 6 Overall experimental results

No.	Result	Number of URLs	Percent (%)
1	True	9,977	91.18
2	False	965	8.82
3	Total	10,942	100

unwanted action such as advertised purposes or suspicious download. Even still exist some false detection rate, accuracy in this step is 93.90 and 92.09% for normal and suspicious group identifying respectively, and total accuracy reach 92.53% since the total of false detection rate is 7.47%.

In the final step, 842 unclustered URLs (Table 3) will be identified as described in Fig. 4b. In this step, three kinds of URLs normal, suspicious, and malicious are identified. The results are summary in Table 5. In that, malicious URLs are detected with highest accuracy at 94.44% in that 11 URLs are recognized as accessed at very high speed. The next highest true identifying rate is for normal URLs detection with 19.20% and lowest is of suspicious since it reaches 67.87%. These results show that identifying between normal and suspicious unclustered URLs is really tougher since behavior of the communication to them is similar with each other. The overall results for whole stages are concluded in Table 6. In that 100% URLs are clustering and identifying with the accuracy reaches at 91.18% and error rate constitutes 8.82%. In that, there is malicious communication to 5 URLs which are not detected or updated by [9] but they are identified by our method.

8 Conclusion

A new novelty method in clustering and identifying automated communication in HTTP environment is proposed in this research. The method is improved from result [3], accordingly, URLs are not just classified into group but also identified and detected by their access purposes. These findings assist network and system administrator clarify the HTTP automated traffic, which are almost unknown to users, from there the internal threats caused by HTTP used program might be inspected early. The method is being independent from payload signatures, which enables the identification of many kinds of automated communication. As a future work, new features are added, that are related to the URLs properties to improve the accuracy in suspicious and normal identifying for unclustered URLs.

References

1. Dietrich CJ, Rossow C, Pohlmann N (2013) CoCoSpot: clustering and recognizing botnet command and control channels using traffic analysis. Comput Netw 57(2):475–486
2. AsSadhan B, Moura JMF (2014) An efficient method to detect periodic behavior in botnet traffic by analyzing control plane traffic. J Adv Res 5(4):435–448
3. Tran MC, Nakamura Y (2016) Classification of HTTP automated software communication behavior using a NoSQL database. IEIE Trans Smart Process Comput 5(2):94–99
4. Oberheide J, Cooke E, Jahanian F (2008) Cloudav: N-version antivirus in the network cloud. In: Proceedings of the 17th USENIX conference on security symposium, pp 91–106
5. Rajab MA, Ballard L, Lutz N, Mavrommatis P, Provos N (2013) CAMP: content-agnostic malware protection. In: Proceedings of 20th annual network and distributed system security symposium (NDSS)
6. Dubuisson M-P, Jain AK (1994) A modified Hausdorff distance for object matching. In: Proceedings of the 12th IAPR international conference, vol 1, pp 566–568
7. Ester M, Kriegel H-P, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the second international conference on knowledge discovery and data mining (KDD-96), pp 226–231
8. VirusTotal. http://virustotal.com/. Last Accessed Feb 2019
9. McAfee Web Gateway. http://www.mcafee.com/us/products/web-gateway.aspx. Last Accessed Feb 2019