An Application for Monitoring and Analysis of HTTP Communications

Manh Cong Tran[1], Minh Hieu Nguyen[2], and Thi Quang Nguyen[1]
[1] Le Quy Don Technical University, 236 Hoang Quoc Viet, Hanoi, Vietnam
[2] Academy of Cryptography Techniques, 141 Chien Thang, Hanoi, Vietnam
Email: manhtc@gmail.com; hieuminhmta@gmail.com; thinq.mta@gmail.com

Abstract —The World Wide Web is the most prevalence information system on the Internet. That makes HTTP protocol becomes attractive target for cybercrimes take it as communication environment to transmit malicious contents or forbidden information such as user private information. This raises the demand for monitoring and analysis of HTTP traffic in network. In this paper, monitoring features are extracted from HTTP basic properties, based on these an application for monitoring and analysis of HTTP communications is proposed. The system will help network and system administrators to early detect threads in HTTP environment by clustering and identifying HTTP traffic. From there, necessary and suitable decides will be acted.

Index Terms—Monitor, analysis, clustering, cybercrime, malicious

I. Introduction

HTTP traffic is generated from many resources, not just from users but also automatically from software (automated software - autoware). HTTP environment would not only be generated from automated software such as anti-virus updaters but could be generated from grayware like adware or unauthorized application and malicious HTTP bot. Malicious requests' structure is similar as that of legitimate normal requests and their traffic merges adequately with each other. Furthermore, in a large private network, huge requests are generated each day. Therefore, the distinction between malicious and normal activities from HTTP traffic is really a big challenge in HTTP communication environment.

Being different from direct Transmission Control Protocol/Internet Protocol (TCP/IP) connections which are connection orientated, HTTP based communication is connectionless protocol, so in order to maintain the updates or to receive commands from server, HTTP based software follows pull method, where automated software (autoware) actively send requests to their servers. However, in details, there are sophisticated differences in the communications behavior of various kinds of autoware to their sites. HTTP malware or C&C channel detection have focused in most of previous researches such as [1]-[3]. However, internal threats within/to a network are also increasing from other suspicious software such as adware or spyware. Because of the flexibility and interoperability of HTTP since everything users need which can be found through web services, its based communication is always allowed in most of network. This raises the demand for monitoring and analysis of HTTP traffic in network. In this paper, an application for monitoring and analysis of HTTP communications in the network is proposed. The application focuses on HTTP traffic capturing, analysis of behavior of Internet access in HTTP environment. In order to overcome the issue of handling huge of traffic each day, a big-data based system proposal is implemented. In that, a combination between MapReduce of Hadoop [4] and the fast-developing and easy for the later development MarkLogic NoSQL database [5] with xQuery supported [6] are suggested for experiment. The method is experimented with real traffic data generated from a university network.

The remainder of the paper is organized as follows. Related work is discussed in Section 2. In Section 3, features extraction and terminology which included autoware communication behaviour analysis and core terminologies are presented. In section 4 is about detailed description of application model which includes algorithms and all components responsible. Section 5 presents applied big data application, the evaluation for proposed method and experiment results. Finally, conclusion and future work are summarized in Section 6.

II. Related Work

Studies about malware detection over network data use multiple approaches. However, it continues to be a serious challenge since operational methods of malware are constantly changing.

Traditional defense mechanisms such as Anti-Virus (AV) products are the most common content-based malware detection techniques. However, Oberheide et al. [7] and Rajab et al. [8] figures out that many undetected malware binaries by using signature-based techniques, and major AV engines just detect only 30% to 70% of recent malwares. Wei Lu *et al*. in [9] and Ke Wang *et al.* in [10] proposed approaches to identify the abnormal payload generated by malware. In which, normal and malicious payloads were analyzed by finding the byte frequency difference between them. However, their research [9], [10] primarily concentrated on detecting IRC-based and TCP-based C&C rather than HTTP based malware. In addition, HTTP based greyware would

Manuscript received January 10, 2018; revised July 5, 2018.
Corresponding author email: manhtc@gmail.com.
doi: 10.12720/jcm.13.8.456-462

generate different payloads at each time to many URLs or resources, so it is more difficult to determine the payload signature for this kind of software communication. In our research, proposed method is behavior based rather than content or payload-based detection.

Overcoming the issue of content-based detection studies, many studies suggest to use network traffic analysis approaches [1]-[3]. In CoCoSpot of Chiristian *et al*. [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. Sung Jin Kim *et al*. [2] proposed HTTP activity set (HAS) analyzer in detection HTTP C&C, in which the discriminative features set (low density and high content variability) for distinguishing C&C flows based on HTTP activity set are found and evaluated. Through that, user-agent is one of the items for HAS. However, attacker tool can modify user-agent by phishing the legitimate content of begin browsers or software. Therefore, the overall accuracy of the method might be degraded. Basil AsSadhan [3] *et al*. proposed a detection method which concentrates in C&C communication analysis and found that it exhibits a periodic behaviour. In [3], a method which applied discrete time series is analyzed to examine the aggregated traffic behaviour in order to detect botnet C&C communication channels' traffic. These researches [1]-[3] focus on detection of botnet communication to C&C servers, but it is confirmed that HTTP based threats not just come from malicious HTTP bots but 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.

Seungwon Shin *et al*. in [11] proposed a framework to detect bot malware at host and network level. At host level, they monitored human-process interactions by using hook technique to capture user mouse's and keyboard's activities. These hook actions might affect users' PC systems. At network level, a simple way to prevent a malware infected PC from sending out information is to prevent all the direct TCP/IP connections from clients. However, allowing HTTP protocol is really leaking holes which might be exploited by HTTP malware. In [11], to overcome this issue, they monitored DNS queries to determine C&C server, but actually, many botnets use hacked domain names and their resources as C&C server. Therefore, the detection method might be insufficient.

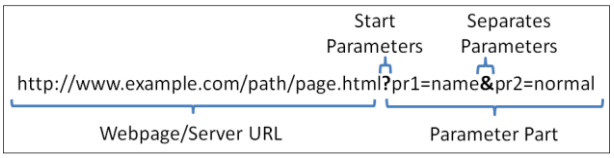

Fig. 1. Main parts of a request

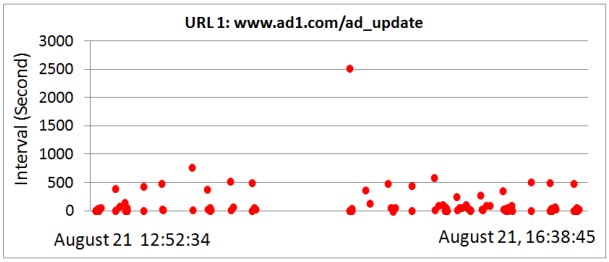

Fig. 2. A sample of an access graph

III. Features Extraction and Terminology

In this paper, the monitoring application focuses on autoware communication behaviour. For that target, by observation of HTTP traffic, autoware communication are analyzed. From which, beneficial features are extracted in order to classify and detect various type of autoware. In this section, background related contents and also core terminologies are presented. HTTP traffic from a client are constituted many requests from that client to outside. At application layer, a request includes these basic information: IP address of client, full URL, Request method. A full URL's parts contain webpage/server URL and parameter path, as shown in Fig. 2. At network level, numerous features are extracted which made from basic client requests information, as follow.

- *Client IP:* Source IP address of machine in network which generated requests
- *Request Method:* main methods of HTTP requests, POST/GET.
- *Request date time:* Date and time when a client send request.
- *Webpage/Server URL (shorten as URL):* URL requested by a Client IP but without parameters part, as shown in Figure 1. Some normal web servers are hacked and some its resource path are exploited as C&C servers. Additionally, parameter part is easily changed base on the specification of requests content, but actually the functionality of that webpage/server URL, such as C&C server or advertise content update, are the same in each request. Therefore, non-parameter URL is used instead of domain or full URL to help more detail and accuracy in classification of autoware access behaviour.
- *Unique URL:* Set of unique URLs requested by a Client.
- *Request Interval:* Break time between two consecutive requests to the same URLs.
- *Request Count:* Number of requests to a URL from a client in a period of observation data.

- *Access Time:* a period of time in seconds which a client accessed to a URL from the fist request to the end request.
- *Access Graph:* Access graph is presented based on the duration of requests to a URL, an access graph is established. This graph presents communication behavior of a client to a specific URL in a specific time period. Assuming that R=(r1, r2,...,rN) is a set of requests from a client to a server/webpage, and all ri have the same webpage/server URL, as illustrated in Fig. 2. In which the X axis is the timing of a request (except the first request) and the Y axis shows the request interval value in seconds. An installed piece of autoware in a client will generate a different access graph for each URL to which it sends requests. For that, this graph can present the behavior in communication between autoware and the webpage or server URLs.

IV. Features Extraction and Terminology

In this paper, based on above features extraction, a big data application is suggested to monitor and analysis HTTP communication. Data for experiment are collected from web proxy of a certain network which served about 2000 clients. Collected data are divided by day saved into logs file as raw data. Big data application is composed by combination of MarkLogic database and MapReduce of Hadoop. In order to clarify traffic in HTTP environment and with the efforts using minor and basic features to solve the problem, many studies in analysis HTTP autoware Internet access behavior has been done. In general, application flows of method are summarized in Fig. 3. In that, HTTP requests are captured and processed through multiple layers and classified and detected into many types of traffic such as normal, suspicious, or malicious.

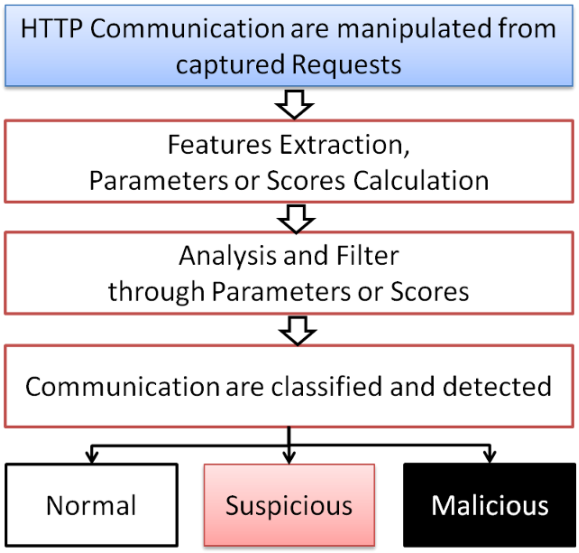

Fig. 3. General flows of proposed methods in this research.

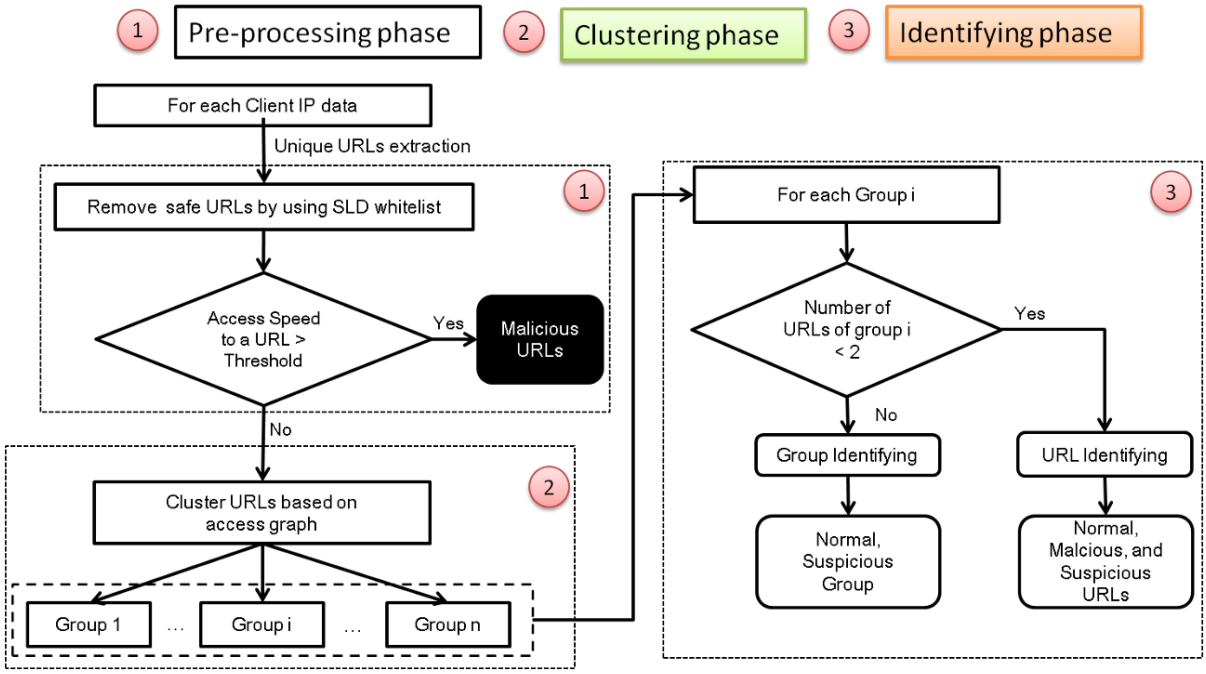

Fig. 4. Phases of monitoring and analysis application.

There are three phases in the application; they are included in Fig. 4. Three phases are pre-processing, clustering and identifying phases. Accordingly, bellow steps will be processed:

- The traffic will be captured from network, for example from a proxy.
- These traffics will be preprocessed in order to reduce the unnecessary processed data by checking the legitimate via a second level domain.
- Next, remain URLs will be group by a unique URLs set without parameter. This set will be clustered in clustering phase based on an algorithm

- After that, in the clustering phase, unique URLs will be classified into two group clustered URLs and unclustered URLs

Hadoop is a great tool to help database application developers and organizations to store and analyze massive amounts of structured and unstructured data from disparate data sources, which data are too massive to manage effectively with traditional relational databases. Hadoop has become popular because it is designed to cheaply store data in the Hadoop Distributed File System (HDFS) and run large-scale MapReduce jobs for batch analysis. MapReduce is a processing framework that uses a divide-and-conquer paradigm that takes a huge task and breaks it into small parts (Map) and then aggregates the resulting outputs from each part (Reduce). Any large task that can be broken into smaller pieces is a candidate for use with Hadoop [4].

The combination between MarkLogic database and MapReduce of Hadoop in this framework is described in Fig. 5. Whereby, a cluster of MarkLogic deveper versioom is set, and due to optimizing performance in query to database, three XML Database Connector (XDBC) application servers, Data Collection, Clustering and Data Analysis, are configured along with a number of forests. There are three modules are working independently for each phase (three phases), details are expressed as bellow:

Data Manipulation Module (DMM) which will read raw log files, convert to XML and text format, and do the preprocessing before stored into Mark-Logic database via Data Collection Application Server. Pre-processing phase (phase 1) is included as a part in this module.

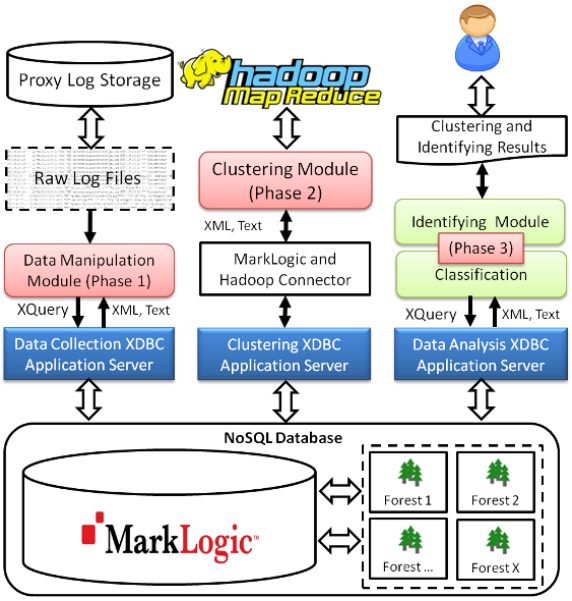

Fig. 5. Big data application for monitoring and analysis HTTP communications.

Core and heavy functions are deployed in the middle part between Mark-Logic database and MapReduce of Hadoop, Clustering Module (CM). This module will archive results from Pre-processing phase, and URLs are clustered in MapReduce by the distributed processing paradigm. This is clustering phase (phase 2). Finally, results of Phase 2 will be returned to MarkLogic database through Clustering XDBC application server. The data exchange between MarkLogic and MapReduce of Hadoop will be undertaken by a connector [12]. Detail process flow of this phase is described in Fig. 6.

TABLE I: EXPERIMENTAL SYSTEM SPECIFICATION

No	Items	Description
1	MarkLogic	A Cluster is installed in two PCs, PC1 is exchange data to MapReduce of Hadoop (phase 2), PC2 is implemented as phase 1 and phase 3
		PC1 specification - CPU: Intel Core 2 Quad CPU Q6600 - CPU Cores: 4 - Processor base operating frequency: 2.4 GHz - RAM: 4GB
		PC2 specification - CPU: Intel Core i7 965 - CPU Cores: 4 - Processor base operating frequency: 3.2 GHz - RAM: 8GB
2	Mapreduce of Hadoop	Execution on a sever with below Specification: - CPU: Intel Xeon Processor E5-2430 v2 (15M Cache, 2.50 GHz) - CPUs: 2 - CPU Cores: 6 - Threads number: 12 - Processor base operating frequency: 2.5 GHz - RAM: 128GB
3	Experiment System Programming	Programming Language are C# on Microsoft DotNet Framework and Java

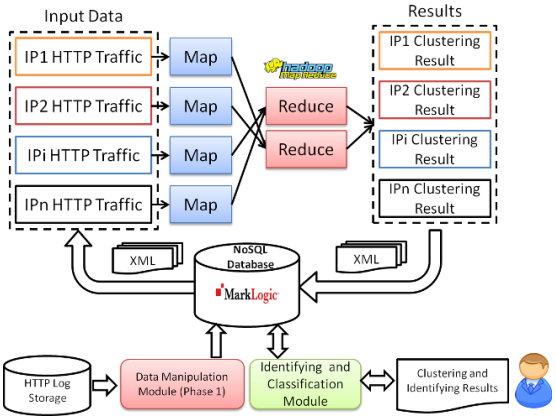

Fig. 6. Process flow of clustering phase on MapReduce of hadoop.

Detection and Identification Module (DIM) is implemented for identifying phase. It will process the result which archived from Phase 2, and work with database through Data Analysis Application Server, after that give out processed results.

Each modules DMM, CM and DIM in framework are implemented as each pre-processing, clustering, and identifying phase respectively. In the experimental environment, MarkLogic Connector for Hadoop 2.6.0 [12] and a free developer licenses of MarkLogic version 8.0.1 [13]. Outbound HTTP traffic from a university network is captured through a proxy server in separated files and stored in a proxy storage.

V. Experiment Results

The monitoring and analysis application is experiment with the captured in an university network. Three phases

Table II: Experimental Data

No	Item	Values
1	Number of Log (Source IPs)	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	Min access time in a log data	150 minutes
6	Max access time in a log data	5 days
7	Number of Requests after Processing Phase.	9,540,608
	Remained percent of total requests	58.85%
8	Number of Unique URLs after Preprocessing Phase	10,942

Table III: Overall Experimental Results

No	Item	Number URLs	Percent
1	TRUE	9,977	91.18%
2	FALSE	965	8.82%
3	Total	10,942	100%

In Fig. 4 are implemented in experiment environment as described in Table I. In order to have balance input data to evaluate the processing time, experimental data of clients Internet access are captured in one day, around 24 hours and summarized in Table II. Details of access time and request number are expressed in Fig. 7 and Fig. 8. In that, number of requests in each log are from 58,606 to 479,751 times. Processing time of application (excluding processing time of phase 1) is shown in Fig. 9. In that, data of clients will be grouped into 5 clients and executed increasingly from 1, 5, 10 and so on until 70 clients. As can be seen that, the processing time fast increased since process data of 1 to 25 clients, after that, the processing time is stable around 18 seconds since the number of clients are in the range from 30 to 70. The processing time takes logarithmic complexity. Overall, minimum and maximum of processing time are respectively 8.56 and 19.18 seconds, average processing time is 15.58 seconds.

All output results are manually check with the support of VirusTotal online system [14] and McAfee Web Gateway [15] which is installed in experiment network. The overall results for whole phases are concluded in Table III. In that 100% URLs are clustering and identifying with the accuracy reaches at 91.18% and error rate constitutes 8.82%.

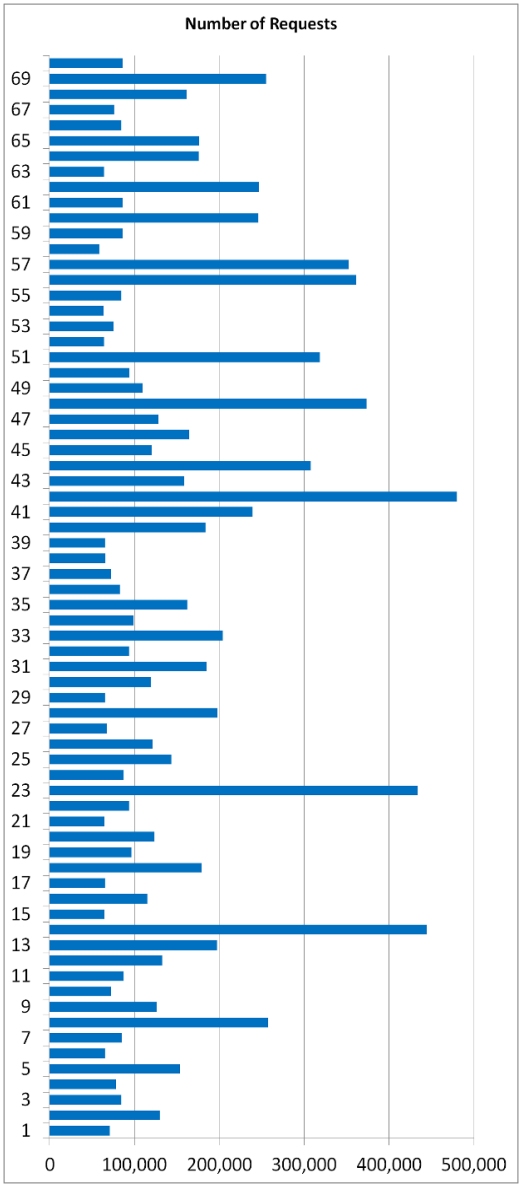

Fig. 7. Number of requests in each log data of one IP. X axis shows the number of request and Y axis show the client's log data index from 1 to 70.

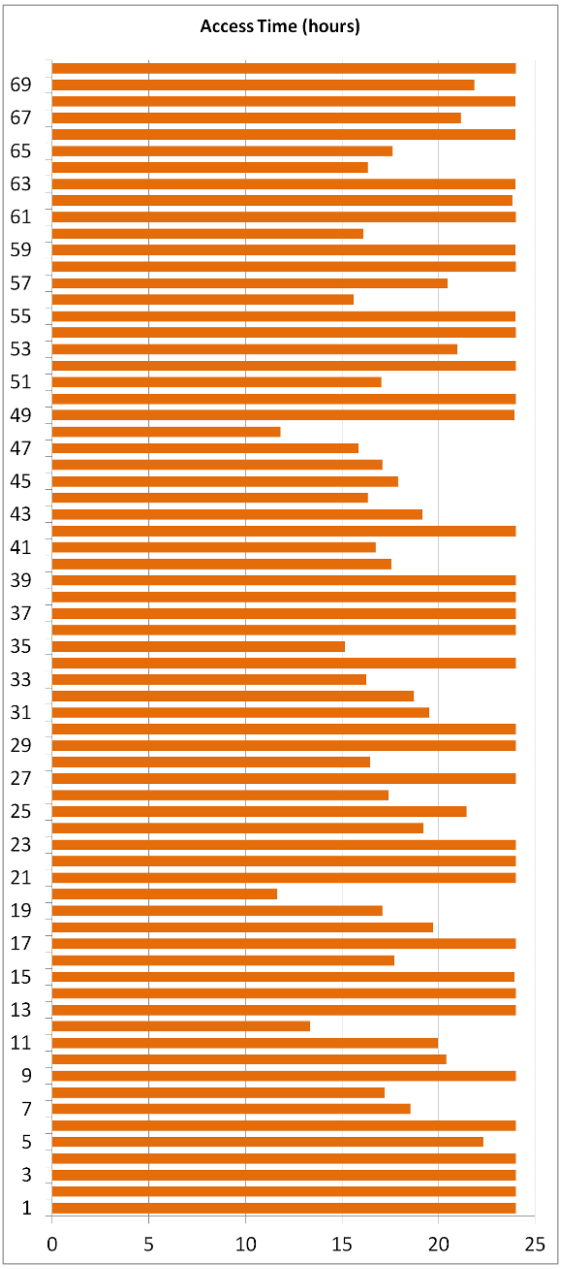

Fig. 8. Access time from client IP to the Internet. X axis shows the access time in Hours and Y axis show the client's log data index from 1 to 70.

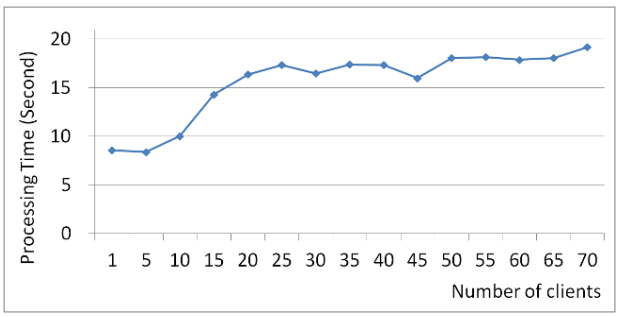

Fig. 9. Processing time of experimental implementation.

VI. Conclusions and Future Work

In this paper, a new novelty monitoring and analysis application in clustering and identifying HTTP communications is proposed. 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 autoware might be inspected early. The huge benefit of the method is being independent from payload signatures which enable the identification of many kinds of automated communication with obfuscated and encrypted message contents. The method works well on any network which just allow HTTP in outbound traffic to prevent threats from TCP/IP direct connection. The accuracy of the approach is equally high and achieve 91.18% for overall and 94.44% for malicious URLs detection by using only a few features at application level. The evaluation also proved that the results are not dependency period of captured data.

The application is experimented by using real traffic and give good results since mixture traffic of all kind HTTP are still identified and detected. The application can be processed distributed, these will help reducing the processing time. In that, the experimental implementation for network level method in a big data system, which are combined MarkLogic NoSQL Database with Map Reduce of Hadoop, proves that it shows good performance since it has to process with huge traffic.

References

[1] C. J. Dietrich, C. Rossow, and N. Pohlmann, "CoCoSpot: Clustering and recognizing botnet command and control channels using traffic analysis," *Computer Networks*, vol. 57, no. 2, pp. 475-486, February 2013.

[2] S. Kim, S. Lee, and B. Bae, "HAS-Analyzer: Detecting HTTP-based C&C based on the analysis of HTTP activity sets," *KSII Transactions on Internet and Information Systems*, vol. 8, no. 5, May 2014.

[3] B. AsSadhan and J. M. F. Moura, "An efficient method to detect periodic behavior in botnet traffic by analyzing control plane traffic," *Journal of Advanced Research*, vol. 5, no. 4, pp. 435-448, 2014.

[4] MapReduce Tutorial, Apache Hadoop. (2008). [Online]. Available:https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

[5] MarkLogic database. What is Marklogic. [Online]. Available: http://www.marklogic.com/what-is-marklogic/

[6] MarkLogic Product Documentation. [Online]. Available: https://docs.marklogic.com/

[7] J. Oberheide, E. Cooke, and F. Jahanian, "Cloudav: N-version antivirus in the network cloud," in *Proc. 17th USENIX Conference on Security Symposium*, 2008, pp. 91-106.

[8] M. A. Rajab, L. Ballard, N. Lutz, P. Mavrommatis, and N. Provos, "CAMP: Content-agnostic malware protection," in *Proc. 20th Annual Network and Distributed System Security Symposium (NDSS)*, 2013.

[9] W. Lu, M. Tavallaee, and A. A. Ghorbani, "Automatic discovery of botnet communities on large-scale communication networks," in *Proc. 4th International Symposium on Information, Computer, and Communications Security (ASIACCS '09)* ACM, New York, NY, USA. Article (CrossRef Link), 2009, pp. 1-10.
[10] K. Wang and S. J. Stolfo, "Anomalous payload-based network intrusion detection," *Recent Advances in Intrusion Detection (RAID) Lecture Notes in Computer Science*, vol. 3224, pp. 203-222, 2004.
[11] S. Shin, Z. Xu, and G. Gu, "EFFORT: A new hostnetwork cooperated framework for efficient and effective bot malware detection," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 57, no. 13, pp. 2628–2642, September 2013.
[12] MarkLogic Connector for Hadoop. [Online]. Available: https://docs.marklogic.com/guide/mapreduce/quickstart
[13] MarkLogic Developer License. Enterprise NoSQL Power for Developers. (2008). [Online]. Available: https://developer.marklogic.com/free-developer
[14] VirusTotal. [Online]. Available: http://virustotal.com/
[15] McAfee Web Gateway. [Online]. Available: http://www.mcafee.com/us/products/web-gateway.aspx

Manh Cong Tran got his master-degree in computer science from Le Quy Don Technical University of Vietnam in 2007. In 2017 Manh got his PhD degree from Department of Computer Science, National Defense Academy, Japan. His current research interests include network traffic classification/analysis and anomaly/malicious detection. Currently, Dr. Manh works as a researcher in Le Quy Don Technical University, Hanoi, Vietnam.

Dr. Minh Hieu Nguyen is a Lecturer with the Academy of Cryptography Techniques (Ha Noi, Viet Nam). He received his Ph.D. from the Saint Petersburg Electrical Engineering University (2006). His research interests include cryptography, communication and network security. He has authored or co-authored more than 65 scientific articles, books chapters, reports and patents, in the areas of his research.

Dr. Thi Quang Nguyen was born in 1980. He received the B.S., M.S. degrees from the Le Quy Don Technical University, Hanoi, Vietnam, in 2004 and 2008, respectively. Thi got his Ph.D. degree in School of Electronics and Information Engineering, Changchun University of Science and Technology from 2015. He currently works as a researcher in Le Quy Don Technical University and his interested research themes focus on blind deconvolution, image processing and pattern recognition.