Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2016, Article ID 2017373, 11 pages
http://dx.doi.org/10.1155/2016/2017373

Research Article

Communication Behaviour-Based Big Data Application to Classify and Detect HTTP Automated Software

Manh Cong Tran and Yasuhiro Nakamura

Department of Computer Science, National Defense Academy, 1-10-20 Hashirimizu, Yokosuka, Kanagawa 239-0811, Japan

Correspondence should be addressed to Manh Cong Tran; manhtc@gmail.com

Received 25 December 2015; Revised 25 March 2016; Accepted 26 June 2016

Academic Editor: Jun Bi

HTTP is recognized as the most widely used protocol on the Internet when applications are being transferred more and more by developers onto the web. Due to increasingly complex computer systems, diversity HTTP automated software (autoware) thrives. Unfortunately, besides normal autoware, HTTP malware and greyware are also spreading rapidly in web environment. Consequently, network communication is not just rigorously controlled by users intention. This raises the demand for analyzing HTTP autoware communication behaviour to detect and classify malicious and normal activities via HTTP traffic. Hence, in this paper, based on many studies and analysis of the autoware communication behaviour through access graph, a new method to detect and classify HTTP autoware communication at network level is presented. The proposal system includes combination of MapReduce of Hadoop and MarkLogic NoSQL database along with xQuery to deal with huge HTTP traffic generated each day in a large network. The method is examined with real outbound HTTP traffic data collected through a proxy server of a private network. Experimental results obtained for proposed method showed that promised outcomes are achieved since 95.1% of suspicious autoware are classified and detected. This finding may assist network and system administrator in inspecting early the internal threats caused by HTTP autoware.

1. Introduction

Application layer attacks pose an ever serious threat to network security for years since it always comes after a technically legitimate connection has been established. Because of the flexibility and interoperability of HTTP since everything users need can be found through web services, its based communication is always allowed in most of network. Consequently, HTTP-based automated software (autoware) is blooming in utilizing in reaching Internet users. Unfortunately, besides normal autoware such as for operating system or software updating purpose, in recent years, cyber criminals turn to fully exploit web as a medium of communication environment to lurk a variety of forbidden or illicit activities through spreading HTTP malicious autoware such as fraudulent adware, spyware, or bot. HTTP traffic and autoware can be classified in some categories as in Figure 1:

(i) Human traffic is kind of traffic which is generated by users with their intention when they use normal software such as web browser to access their websites to get information they needed. In this kind of traffic, users clearly understand their accessed sites, who they contact to, and which information they obtain.

(ii) On the other side, the graph presents nonhuman traffic to which users unintentionally have access; they come from autoware. This traffic can be requested from normal software such as antivirus updater, mail client, browser's toolbar, greyware encompasses adware, spyware, joke programs, and malicious software acting as HTTP-based botnet and trojan horses.

Normal autoware can be controlled and beneficial for user; however, since greyware and malicious software penetrate into users' network, they turn out to be internal threats, from which attackers can conduct various types of application layer attacks through these agents, which are really difficult to prevent such as DoS/DDoS, malware distribution, or identity theft. The distinction between malicious and normal activities from HTTP traffic is becoming tougher because the malicious requests merges adequately with legitimate HTTP

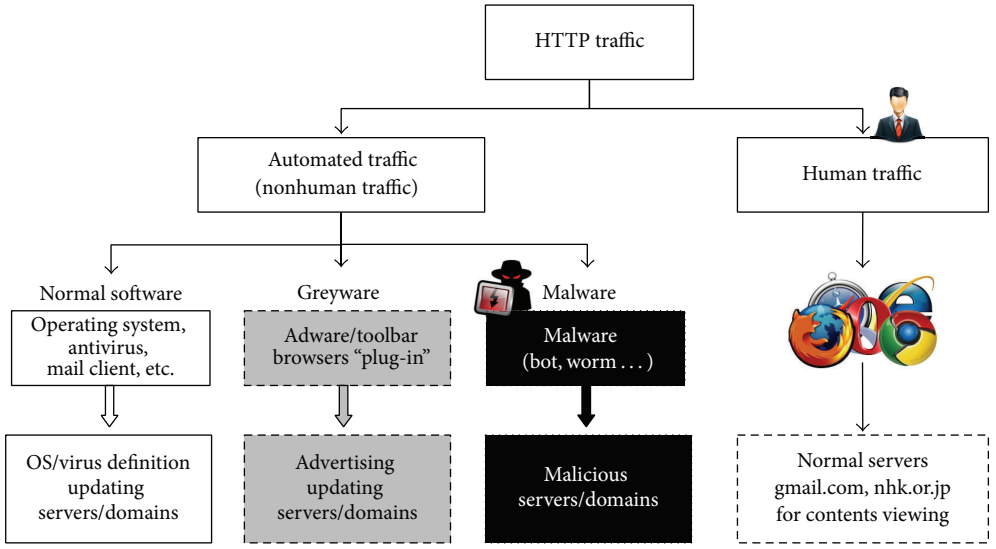

Figure 1: HTTP traffic and automated software categories.

traffic. Furthermore, in a large private network, detection and also classification between types of HTTP autoware traffic are really great challenge when huge requests are generated each day.

To maintain communication, perform updates, or receive commands, all kinds of HTTP-based autoware have common characteristics that they generate repetitively legal traffic and requests to their servers/domains. However, in detail, there are some sophisticated differences in the way of communication behaviour of autoware to their sites. In this paper, based on the analysis and study of autoware communication behaviour, a method in classification and detection of HTTP autoware at network level is proposed. 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 [1] and MarkLogic NoSQL database [2] with xQuery supported [3] is suggested for experiment. The method is experimented with real traffic data generated from a university network, and a promised result is archived in classification and detection of malicious HTTP autoware communication.

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. Section 4 is about detailed description of proposed method 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.

2. Related Work

There were a considerable number of techniques which aim to protect users against malware; however, it continues to be a challenging problem. Traditional defense mechanisms such as antivirus (AV) products are the most common content-based malware detection techniques. These types of AV software run on end-user systems and employ signature-based detection to identify variants of known malware. As a consequence, the signature generation and update cycle cause an inherent delay in protecting users against new variants of malware [4]. Additionally, with the aim of limiting AV engines effectiveness, malware authors have developed increasingly sophisticated evasion techniques such as packing and polymorphism, aimed at circumventing detection by AV engines [5, 6]. Oberheide et al. [7] figure many undetected malware binaries by using signature-based techniques, and major AV engines just detect only 30% to 70% of recent malware. As the same content, Rajab et al. [4] show that less than 40% of malicious binaries can be detected by four AV engines in their experiment.

Many botnet detection methods are presented in [8–11]. Ashley [8] has suggested a method for detecting potential HTTP C&C activity based on repeated HTTP connections to a website. According to this, an algorithm is proposed for detecting HTTP polling activity. Lu et al. in [9], using signature-based techniques, propose a hierarchical framework to automatically discover malicious bot on a large-scale Wi-Fi ISP network, in which the network traffic is classified into different application communities by using payload-signature. These signatures were used to separate known traffic from unknown traffic in order to decrease the false alarm rates. Eslahi et al. [10] proposed an approach to reduce the false alarm HTTP botnet detection; in this research, high access rate traffic, which might be other security threats, is filtered out. Basil AsSadhan and Moura [11] proposed a detection method in which it concentrates in C&C communication analysis and find that it exhibits a periodic behaviour. In [11], a method which applied discrete time series is analyzed to examine the aggregate traffic behaviour in order to detect botnet C&C communication channels traffic. These researches [8–11] focus on botnet communication to C&C server, but

actually HTTP threats do not just come from malicious bots but also can be from other types of automated software such as HTTP spyware, adware, or unauthorized applications.

Shin et al. in [12] proposed a framework to detect bot malware at host and network level. At host level, they monitor human-process interactions by using hook technique to capture user mouse and keyboard activities. These hook actions might affect users PC systems. At network level, a simple way to prevent a malware infected PC sending out the information is to prevent all the direct TCP/IP connection from clients. However allowing HTTP protocol is really leaking hole which might be exploited by HTTP malware. In [12], to overcome this issue, they monitored DNS queries to determine C&C server, but actually, many botnets use hacked URL as C&C server. Therefore, the detection method might be insufficient.

Some of approaches use lexical features or keywords extracted from URL and web contents as in [13–16]. However, many other types of malicious web pages are disguised by domain names or URLs like normal website and can harm users PC systems. In this case, lexical or keywords features might be compromised. Bartlett et al. [17] proposed an approach to identify low-rate periodic network traffic and changes in regular communication of autoware. Their research also focuses on many types of autoware and monitor TCP flows to detect, but, in this paper, the target does not just focus only on detecting general types of autoware but also on particular URLs where autoware request to. In addition, our method just collects and processes with basic features of HTTP Traffic at application layer. This will help reduce process cost compared with method used TCP packets features since the number of packets to be processed increases.

3. Features Extraction and Terminology

In this paper, classification and detection method is based on autoware communication behaviour. For that target, by observation of HTTP traffic, autoware communication is analyzed, from which beneficial features are extracted in order to classify and detect various types of autoware. In this section, background related contents and also core terminologies are presented.

3.1. Features Extraction. HTTP traffic from a client consisted of many requests from that client to outside. At application layer, a request includes basic information: IP address of client, full URL, and request method. Full URL's parts contain webpage/server URL and parameter path, as shown in Figure 2. At network level, numerous features are extracted which are made from basic client requests information as follows:

(i) Client IP: source IP address of machine in network which generated requests.

(ii) Request method: main methods of HTTP requests, POST/GET.

(iii) Request date time: date and time when a client sends request.

(iv) Webpage/server URL (shorten as URL): URL requested by a client IP but without parameters' part, as shown in Figure 2. Some normal web servers are hacked and some of their resource paths are exploited as C&C servers. Additionally, parameter parts are easily changed based on the specification of requests content, but actually the functionality of that webpage/server URL, such as C&C server or advertise content update, is the same in each request. Therefore, nonparameter URL is used instead of domain or full URL (will be parameter part), and this matter will help the classification of autoware access behaviour become more detailed and accurate.

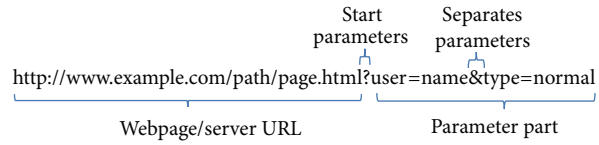

Figure 2: Main parts of URL.

(v) Unique URL: set of unique URLs requested by a client.

(vi) Request interval: break time between two consecutive requests to the same URLs.

(vii) Request count: number of requests to URL from a client in a period of observation data.

(viii) Access time: a period of time in seconds during which a client accessed to URL from the fist request to the end request.

3.2. Access Graph. Access graph presents communication behaviour of a client to a specific URL in a duration of time. It is formed on request interval which are extracted from HTTP traffic. Assuming that $R = \{r_1, r_2, \ldots, r_N\}$ is set of requests from a client to a webpage/server and all r_i have the same webpage/server URL, as described in Figure 2, then access graph G is a sequence which included $N - 1$ items, $G = \{g_1, g_2, \ldots, g_{N-1}\}$, where g_i is a pair of (t_i, d_i), where t_i is timing of request r_{i+1} and d_i is request interval between r_i and r_{i+1}. An access graph is shown as in Figure 3, in which, X-axis is timing of request (except the first request) and Y-axis shows the request interval value in second. An installed or infected autoware client will establish a different access graph for each URL which it sends requests to. For that, this graph can present the behaviour in communication between an autoware to its webpage or server URLs.

3.3. Autoware Communication Behaviour. For keeping communication, update or receive command, all kinds of HTTP-based autoware have common characteristics that they generate repetitively legal traffic and requests to their servers/domains. However, in detail, there are some sophisticated differences in the way of communication behaviour of autoware to their sites.

(i) Malicious HTTP-based bots always follow the PULL style where they connect to their command and control server periodically in order to get the commands and updates. The number of requests from malicious bots are not high as normal autoware (e.g., updater

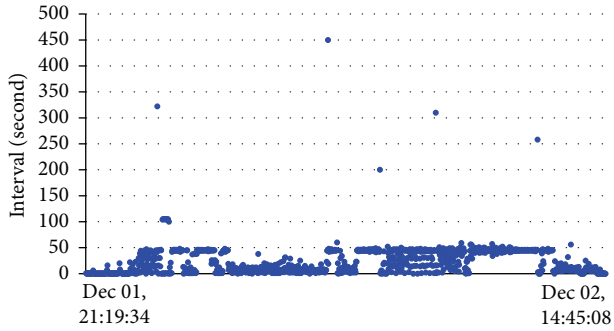

Figure 3: An access graph of a client request to URL.

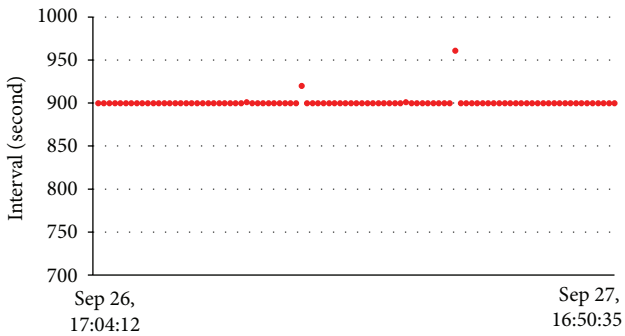

Figure 4: An access graph of HTTP malicious bot.

and downloader) which just generate requests with a long interval than unusual malicious bots [10, 11, 18]. Because interval in communication between a malicious bot to their C&C server is stable, there is almost no variation in their access graph as can be seen in Figure 4 showing the access graph of a bot communication.

(ii) Malicious bots often connect to one control domain and to a specific server resource. Difference with that, unwanted HTTP applications, or greyware, such as annoying adware or spyware, often report back to or request new information from many external resources [17]. Therefore, they keep communicating to their numerous advertising sites or URLs to update pop-up or advertisement and commercial content areas. Autoware will behave the same communication pattern to its URLs if they are requested at the same or approximately equivalent timing so access graph of URLs from a specified autoware is looked similar. In addition, many URLs are requested with the same timing by a specified autoware, so the access duration to these URLs is approximately equal. It means that the first and the last requests timing to these URLs are the same with others. In Figure 5, a sample of two similar access graphs presents the communication from one autoware to two different URLs, and the first and the last requests moment of them are equal.

(iii) On the contrary with autoware, there are no interval or periodic patterns in users' web access; however, in recent years, many sites (e.g., shopping online site or social media webpage) append advertisement path to their sites and use JavaScript or Flash as autoaware part to automatically collect the advertising content as adware or spyware. Therefore parts of users access sites can generate HTTP traffic which act as autoware communication.

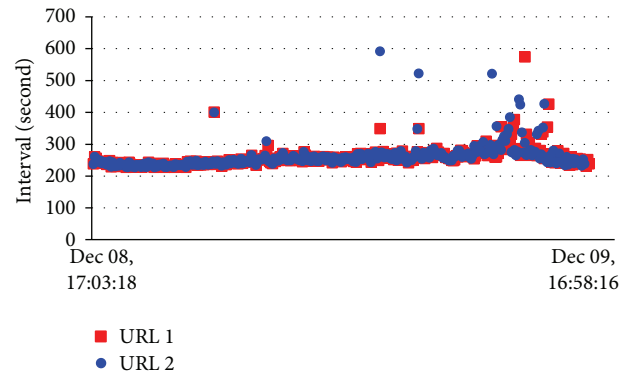

Figure 5: Access graphs from an autoware of a client IP to two different URLs are similar, and the access times of URLs are the equal since both of them are requested from Dec 08, 17:03:18, to Dec 09, 16:58:16.

3.4. Access Graph Distance. As analysis in Section 3.3, even URLs are different; if they are requested by the same autoware then the access graphs look similar, as can be seen in Figure 5. This part proposes a distance to measure the similarity of autoware access behaviour in communication to URLs from a client. The calculation method is based on Modified Hausdorff (MH) distance which is presented in [19].

Assume that there are two access graphs $A = (a_1, \ldots, a_N)$ and $B = (b_1, \ldots, b_M)$. Define that the distance between two points a_i and b_j is calculated as Euclidean distance $d(a_i, b_j) = \|a_i - b_j\|$. From that, distance between point a_i and graph B is defined as $d(a_i, B) = \min_{b_j \in B} \|a_i - b_j\|$. Generalized Hausdorff distance of A and B in [19, 20] is defined as follows:

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

Based on (1), distance between access graphs A and B, which follow by MH distance (MHD), is formed as follows:

$$\mathrm{MHD}(A, B) = \max(d(A, B), d(B, A)). \quad (2)$$

The smaller the MH distance between A and B is, the more A and B are similar to each other.

3.5. Suspicious Score. As described in Section 3.3, malicious bots connect to their command and control server (C&C server) periodically in order to get the commands and updates; therefore, almost there is no large variation in the access graph from malicious bot to its C&C, as can be seen in Figure 4. Based on this analysis, a score is proposed to measure the variation of a access graph, from which it shows suspicious of communication between client to its URL.

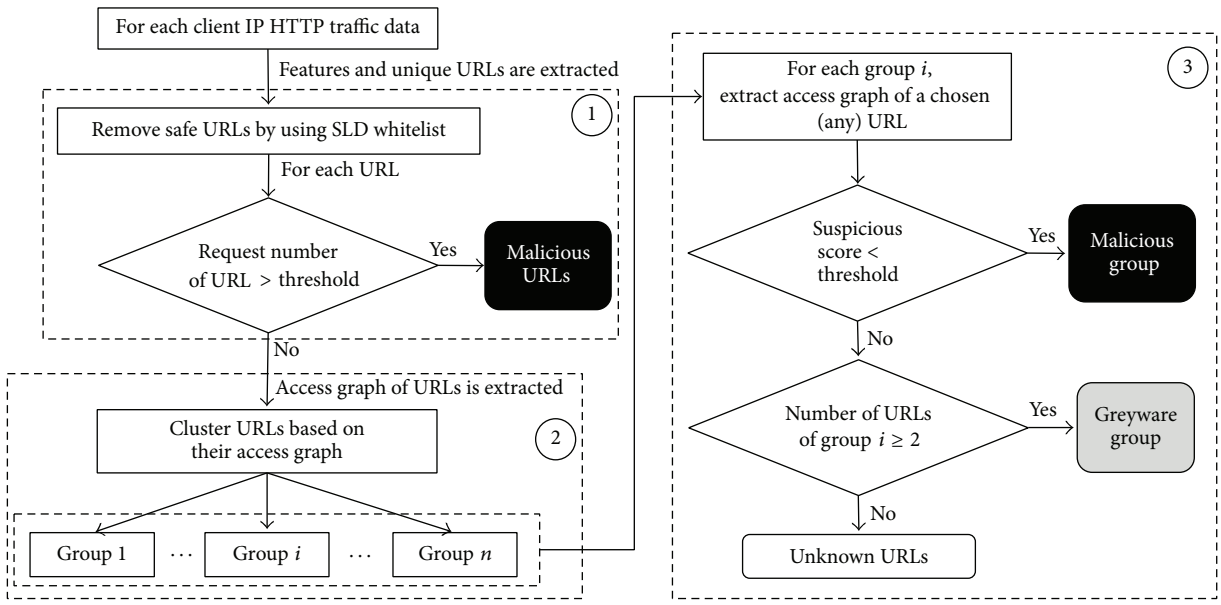

Figure 6: Proposed method diagram in classification and detection of HTTP automated software. Labels of 1, 2, and 3 are preprocesing, clustering, and detection/classification phase, respectively.

Assuming that the access graph of URL S is specified and denoted as $X = (x_1, \ldots, x_N)$, a suspicious score will be defined as *coefficient of variation* of X as follows:

$$\text{Suspicious Score}(X) = \frac{\sigma}{\mu} \tag{3}$$

in which σ and μ are standard deviation and mean of X, respectively. The smaller suspicious score shows that URL is more suspicious.

Figure 7: Phishing websites are less consistent with their content when compared with those of legitimate websites.

4. Proposed Method

Based on the autoware communication behaviour which is described in Section 3 and the observation of access graphs in Section 3.1, a classification and detection method, including three phrases, is proposed as in Figure 6; details are as follows.

4.1. Preprocessing Phase. This preprocessing phase is objective to eliminate unnecessary processed data. For each client IP, the one-day HTTP traffic features are extracted and preprocessed; in order to process this phase two methods are applied:

(i) The first one is to filter URLs requests from client IP through a whitelist of second level domain names (SLDN). This filter method is described in [13]; according to that, the tokens in the URLs of phishing websites are less consistent with their content when compared with those of legal websites. An example is illustrated in Figure 7. In this example, the legitimate website contains the brand names *apple* in the SLDN. Even though the phishing website also contains the brand name *apple* in the URL, it is not in the SLDN. Therefore, a domain name which contains a second level domain name which is defined in SLDN whitelist is marked as benign.

(ii) The second method is based on the number of requests to URL from a client IP. Based on the observations number of requests from autoware to URL, it can be seen that suspicious autoware has access many times to URL in a duration of time. Therefore, if the number of requests to URL is too small, it seems not to be requested by an autoware.

Also in this phase, URLs which are requested with extremely fast speed in a duration time will pose a malicious autoware communication; access speed is defined as follows:

$$\text{Access Speed}(\text{URL}_i) = \frac{\text{Request Count}(\text{URL}_i)}{\text{Access Time}(\text{URL}_i)}. \tag{4}$$

In that access time and request count features are described in Section 3.1.

4.2. Clustering Phase. After preprocessing phase, in this phase, remaining URLs will be clustered into number of

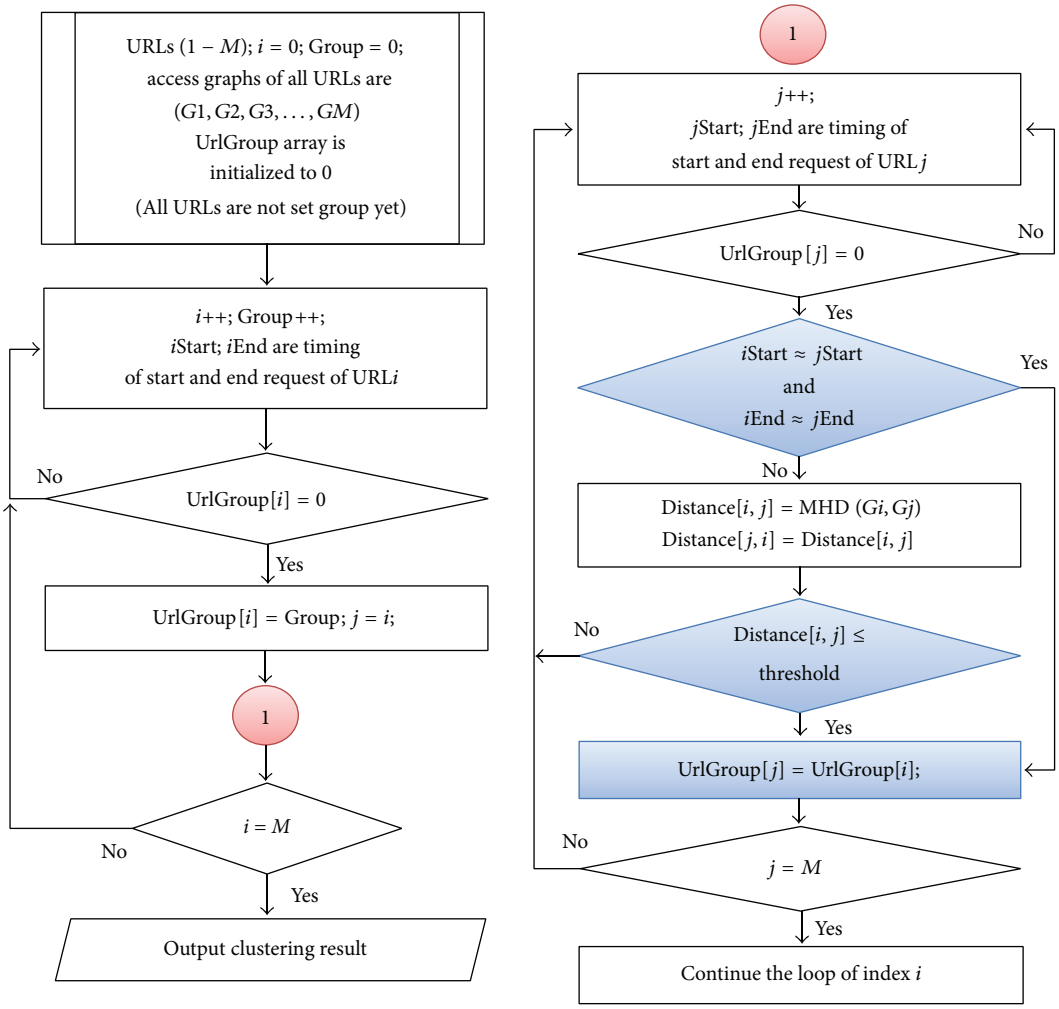

Figure 8: Autoware communication clustering algorithm.

groups based on their characteristics which are presented in Section 3.3. Accordingly, two URLs are of the same group (requested by the same autoware from a client) if they match one of following conditions:

(i) The first and the last request timing to two URLs are approximately the same.

(ii) Based on the similarity of its access graph, MH distance between two access graphs of URLs is calculated; if this distance is small enough, they will be recognized as in the same group.

An algorithm is suggested to decide a group for any two URLs. In order to optimize the consumption processing time of method, the steps of algorithm are proposed in Figure 8. By using a group label array, from this algorithm, distance between all pair of access graphs need not to be calculated. If URL is labeled to a group, it will not need to check group again with other URLs.

4.3. Detection and Classification Phase. The third phase is detection and classification. For each group, a URL (any in the group) is chosen and its access graph is extracted. Then the suspicious score of this URL is calculated; in order to detect whether it is malicious or not a threshold is proposed as 0.04. If the suspicious score is less than or equal to the threshold it is detected as malicious. Finally, remaining groups will be detected by examining the number of unique URLs in group. As analyzed in Section 3, difference with malicious bots, greyware commonly access to various URLs instead of only one server or URL. Therefore, a group having number of unique URLs which are not less than 2 will be marked as greyware groups.

5. Big Data Proposed Framework and Experiment Results

5.1. Big Data Proposed Framework. In this paper, based on above proposed method, big data application is suggested to classify and detect autoware 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

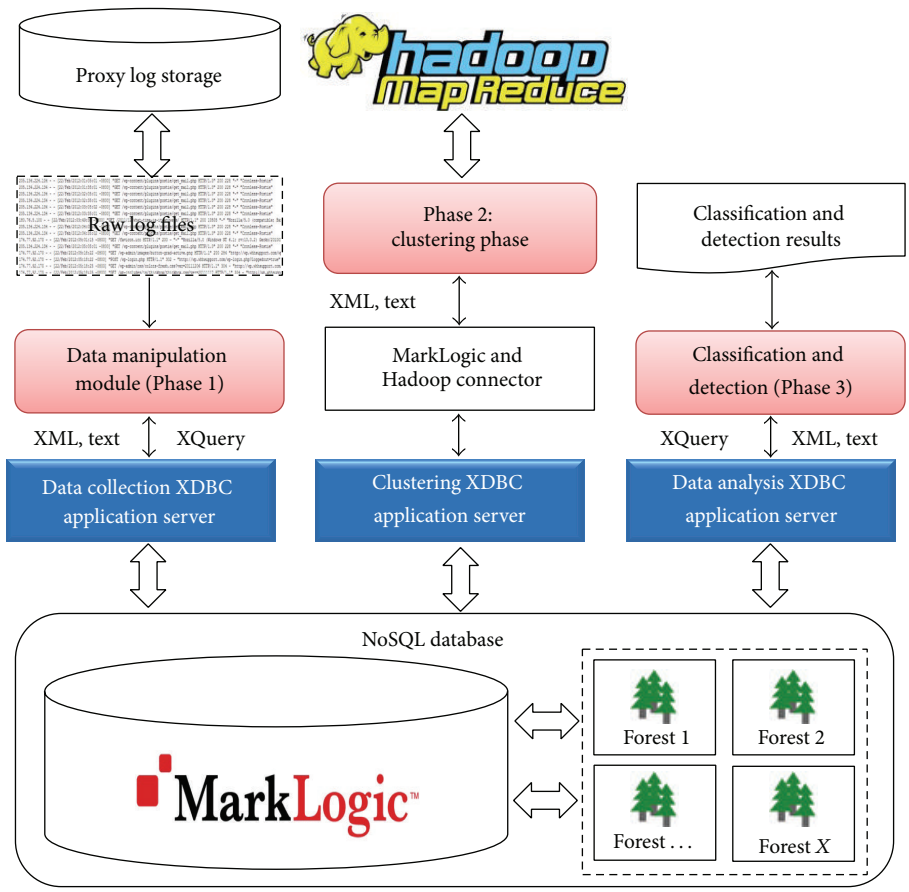

Figure 9: Big data based framework proposal.

composed by combination of MarkLogic database and MapReduce of Hadoop.

As described in [2, 21], MarkLogic is an enterprise NoSQL (Not only Structured Query Language) database which supports a very flexible and convenient XQuery when working with structured and also unstructured data. In addition, it also has had ACID transactions (ACID stands for Atomicity, Consistency, Isolation, and Durability). In a transactional application ACID's properties are necessary so that reads and writes are durably logged to disk and strongly isolated from other transactions. Without this feature, users run the risk of encountering data corruption, stale reads, and inconsistent data. In this framework, XML and text data format are suggested to use because of easily transforming from raw data log file into database.

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, of 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 [2].

The combination between MarkLogic database and MapReduce of Hadoop in this framework is described in Figure 9, whereby a cluster of MarkLogic is set, and due to optimizing performance in query to database, three XDBC application servers, Data Collection, Clustering, and Data Analysis, are configured along with a number of forests. There are three modules working independently for each phase in Figure 6; details are expressed as follows:

(i) Phase 1 is processed as a part in Data Manipulation Module which will read raw log files, convert to XML and text format, and do the preprocessing before being stored into MarkLogic database via Data Collection Application Server.

(ii) Core functions of heavy Phase 2, Clustering Phase, are implemented according to algorithm in Figure 8 and deployed in the middle part between MarkLogic database and MapReduce of Hadoop. This module will archive results from Phase 1, and URLs are clustered in MapReduce by the distributed processing paradigm. 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

Table 1: Experimental data statistic.

Item	Statistic	Unit	Note
Number of logs	95	PC	Log equals HTTP traffic in a day of IP
Total of requests	13,905,165	Request	All requests of 95 logs
Max requests	479,751	Request	Requests from log
Min requests	22,305	Request	
Average requests	146,370	Request	
Max access time	24	Hour	From the first request to the last request
Min access time	6	Hour	
Average requests	20	Hour	

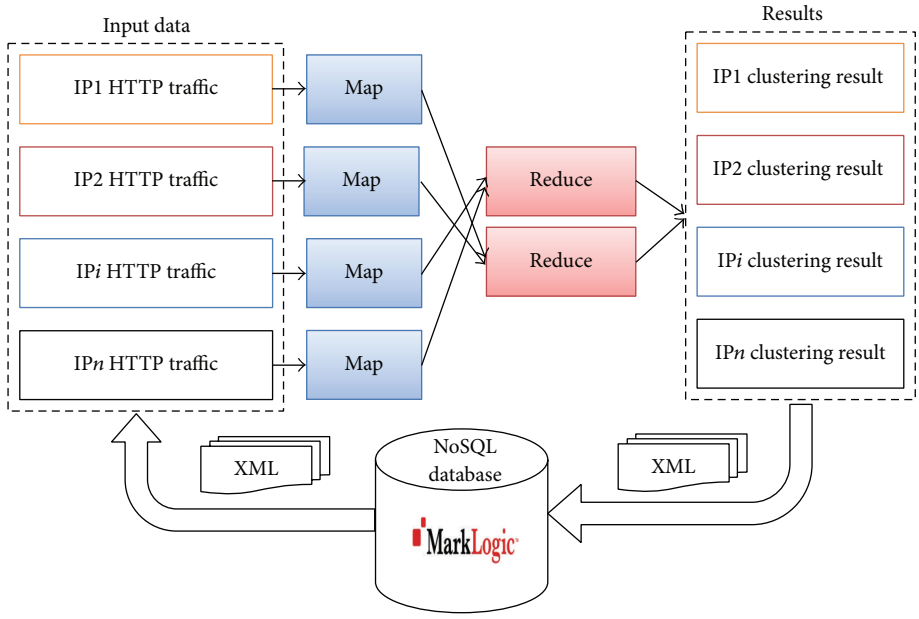

Figure 10: Process flow of clustering phase.

undertaken by a connector. Detailed process flow of this phase is described in Figure 10.

(iii) Classification and Detection Module is implemented for Phase 3, Detection and Classification Phase. It will process the result which is archived from Phase 2 and work with database through Data Analysis Application Server and after that give out processed results.

5.2. Experimental Analysis and Results. Experiment environment is shown in Figure 11; in that free developer licenses of MarkLogic verion 8.0.1 and Hadoop 2.6.0 are used [22]. From this experiment model, HTTP traffic from a university network is captured through a proxy server in separated files which are divided by date and stored in a proxy storage. These logs' raw data files will import to system through Data Manipulation Module as in Figure 9. Denoted log is HTTP traffic of IP in one day, which will be stored in its own directory in MarkLogic; 95 logs' data of clients are extracted, analyzed, and classified through the proposed method. Experiment data is detailed and summarized in Table 1. In that there are two Zeus bots [23] which are installed into a client with difference interval in communication to C&C. All output results are manually checked with the support of VirusTotal online system [24] and McAfee Web Gateway which is installed in experiment network [25].

After preprocessing phase of proposed method described in Figure 6, a set of unique URLs (for logs of each IP) is established with 5621 URLs. In that, there are 14 URLs requested by numerous IPs which are generated with extreme speed over a threshold which is set as 0.8 in this experiment. In Table 2, details of 14 malicious URLs detected by preprocessing phase are summarized. The request per second (access speed) is determined by request count and access time via (4). Based on the characters of malicious autoware which is infected into client IP, the access speed and also communication behaviour to these URLs are determined. For example, as can be seen in Table 2, just in only 0.6 hours, URL2 is requested 80,903 times so it owns highest access speed at 32.98 requests per second. Vise versa, with URL12, it is requested with lowest speed at 0.82 requests per second, 71,004 times in 24 hours; however it is still higher than access speed to other URLs in experimental data. By manually checking the support of [24, 25], all these 14 URLs from domains/web servers contain unwanted software

Table 2: Malicious URLs detected in Phase 1 (preprocessing phase).

Number	Malicious URL	Requests		Access time (h)	Requests per second
		Count	Percent		
1	URL1	237,291	1.71%	2.04	32.38
2	URL2	80,903	0.58%	0.68	32.98
3	URL3	80,032	0.58%	24.00	0.93
4	URL4	303,633	2.18%	10.56	7.98
5	URL5	81,256	0.58%	24.00	0.94
6	URL6	149,966	1.08%	12.53	3.32
7	URL7	496,781	3.57%	4.40	31.39
8	URL8	364,809	2.62%	11.69	8.67
9	URL9	80,761	0.58%	24.00	0.93
10	URL10	297,938	2.14%	16.65	4.97
11	URL11	80,423	0.58%	24.00	0.93
12	URL12	71,004	0.51%	24.00	0.82
13	URL13	80,549	0.58%	24.00	0.93
14	URL14	81,040	0.58%	24.00	0.94
	Total	*2,486,386*	*17.88%*		

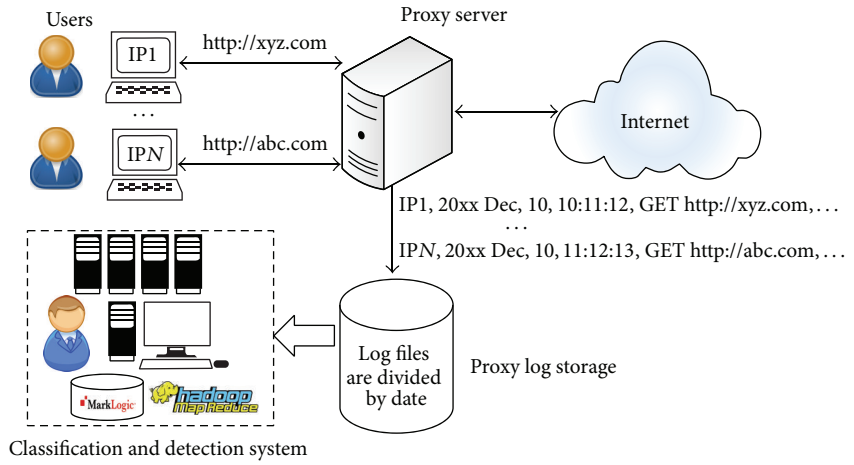

Figure 11: Experiment environment.

and are marked as malicious by many network security companies and software. These 14 URLs are requested 2,486,386 times, and they derive 17.88% of 13,905,165 total requests in experimental data.

Remaining 5607 URLs are classified in 673 groups in which 393 groups which contain 2 URLs above are detected as greyware. MapReduce just needed about 30 seconds to process all these URLs of 95 logs. As results summarized in Table 3, beside 14 malicious URLs which are detected in Phase 1 (preprocessing phase), 5 URLs requested are detected as malicious in Phase 3 (classification and detection phase), 2 of them are matched with C&C servers communicated by installed Zeus bots and other 3 URLs are detected from experimental captured data. All the detected greyware communication groups are confirmed when they come from shopping sites, social media, and adverting companies. Remaining 275 URLs are unclustered; system can not detect these URLs.

Table 3: Experimental results.

Phase	Malicious URLs	Greyware		Unknown URLs
		Group	URLs	
Phase 1	14			
Phase 3	5	393	5327	275

These constitute a false negative of 4.9% and the accuracy rate reaches 95.1%.

6. Conclusion and Future Work

In this paper, a new method is proposed to detect and classify autoware communication based on its behaviour via analysis of HTTP traffic. The major advantage of the proposed method is that it just used minor features in HTTP

traffic and does not use any signature or content-based technique. In addition, big data application framework also is proposed by combination of two leading technologies, which are the power of distributed processing of MapReduce of Hadoop and the convenient in working with unstructured data through XDBC servers of NoSQL database MarkLogic. Experiment results are promised and methods are working well in private network environment.

There are some reasons contributing undetected rate. First, even autoware commonly communicates with sites by the same behaviour, some rare cases of autowares' requests are different. Second, some types of autoware have less activities in network since they just send out little requests. In other situations, users' Internet accessed traffic also might be automated communication since their access sites automatically refresh its contents via HTML script such as JavaScript or Flash. In these cases, clustering and detection of these URLs access graphs are become tougher. Based on this result, with the objective of reducing the undetected rate, some new features need to be considered in the future work. For that matter, data size sent in each request is regarded since this feature from malicious bot communication to its C&C server is almost steady whist variation of adware's data size in each request depends on the content which they get. In addition, unclustered URLs are also considered to be classified by checking the matching between domain name part of them and clustered group which is in clustering phase.

Competing Interests

The authors declare that there are no competing interests regarding the publication of this paper.

References

[1] MapReduce Tutorial, *Apache Hadoop*, 2008, https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html.

[2] MarkLogic database, "What is Marklogic," 2015, http://www.marklogic.com/what-is-marklogic/.

[3] MarkLogic 8 Product Documentation, https://docs.marklogic.com/.

[4] M. A. Rajab, L. Ballard, N. Lutz, P. Mavrommatis, and N. Provos, "CAMP: content-agnostic malware protection," in *Proceedings of the Network and Distributed Systems Security Symposium (NDSS '13)*, Internet Society, 2013.

[5] A. Averbuch, M. Kiperberg, and N. J. Zaidenberg, "An efficient VM-based software protection," in *Proceedings of the 5th International Conference on Network and System Security (NSS '11)*, pp. 121–128, IEEE, Milan, Italy, September 2011.

[6] P. Royal, M. Halpin, D. Dagon, R. Edmonds, and W. Lee, "PolyUnpack: automating the hidden-code extraction of unpack-executing malware," in *Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC '06)*, pp. 289–298, IEEE, Miami Beach, Fla, USA, December 2006.

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

[8] D. Ashley, *An Algorithm for HTTP Bot Detection*, University of Texas at Austin—Information Security Office, Austin, Tex, USA, 2011.

[9] W. Lu, M. Tavallaee, and A. A. Ghorbani, "Automatic discovery of botnet communities on large-scale communication networks," in *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security (ASIACCS '09)*, pp. 1–10, ACM, Sydney, Australia, March 2009.

[10] M. Eslahi, H. Hashim, and N. M. Tahir, "An efficient false alarm reduction approach in HTTP-based botnet detection," in *Proceedings of the IEEE Symposium on Computers & Informatics (ISCI '13)*, pp. 201–205, Langkawi, Malaysia, April 2013.

[11] 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.

[12] S. Shin, Z. Xu, and G. Gu, "EFFORT: a new host-network cooperated framework for efficient and effective bot malware detection," *Computer Networks*, vol. 57, no. 13, pp. 2628–2642, 2013.

[13] Y.-S. Chen, H.-S. Liu, Y.-H. Yu, and P.-C. Wang, "Detect phishing by checking content consistency," in *Proceedings of the 15th IEEE International Conference on Information Reuse and Integration (IRI '14)*, pp. 109–119, Redwood City, Calif, USA, August 2014.

[14] A. Blum, B. Wardman, T. Solorio, and G. Warner, "Lexical feature based phishing URL detection using online learning," in *Proceedings of the 3rd ACM Workshop on Artificial Intelligence and Security (AISec '10)*, pp. 54–60, 2010.

[15] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious URLs," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*, pp. 1245–1254, ACM, Paris, France, July 2009.

[16] T.-C. Chen, S. Dick, and J. Miller, "Detecting visually similar web pages: application to phishing detection," *ACM Transactions on Internet Technology*, vol. 10, no. 2, article 5, pp. 5:1–5:38, 2010.

[17] G. Bartlett, J. Heidemann, and C. Papadopoulos, "Low-rate, flow-level periodicity detection," in *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS '11)*, pp. 804–809, April 2011.

[18] M. C. Tran and Y. Nakamura, "In-host communication pattern observed for suspicious HTTP-based auto-ware detection," *International Journal of Computer and Communication Engineering*, vol. 4, no. 6, pp. 379–389, 2015.

[19] M.-P. Dubuisson and A. K. Jain, "A modified Hausdorff distance for object matching," in *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Conference A: Computer Vision & Image Processing*, vol. 1, pp. 566–568, IEEE, Jerusalem, Israel, 1994.

[20] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing images using the Hausdorff distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 850–863, 1993.

[21] C. Brooks, *Enterprise NoSQL for Dummies*, John Wiley & Sons, Hoboken, NJ, USA, 2014.

[22] MarkLogic Developer License, *Enterprise NoSQL Power for Developers*, 2008, https://developer.marklogic.com/free-developer.

[23] N. Falliere and E. Chien, "Zeus: King of the bots," Symantec Security Response, 2009, https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/zeus_king_of_bots.pdf.

[24] VirusTotal, 2015, http://virustotal.com/.

[25] McAfee Web Gateway, http://www.mcafee.com/us/products/web-gateway.aspx.