IEIE Transactions on Smart Processing and Computing, vol. 5, no. 2, April 2016
http://dx.doi.org/10.5573/IEIESPC.2016.5.2.094

Classification of HTTP Automated Software Communication Behavior Using a NoSQL Database

Manh Cong Tran and Yasuhiro Nakamura

Department of Computer Science, National Defense Academy, 1-10-20 Hashirimizu, Yokosuka, Kanagawa, Japan
manhtc@gmail.com, yas@nda.ac.jp

* Corresponding Author: Manh Cong Tran

Received February 20, 2016; Accepted April 19, 2016; Published April 30, 2016

** Extended from a Conference: Preliminary results of this paper were presented at the ICEIC 2016. This present paper has been accepted by the editorial board through the regular reviewing process that confirms the original contribution.*

***Abstract*:** Application layer attacks have for years posed an ever-serious threat to network security, since they always come after a technically legitimate connection has been established. In recent years, cyber criminals have turned to fully exploiting the web as a medium of communication to launch a variety of forbidden or illicit activities by spreading malicious automated software (auto-ware) such as adware, spyware, or bots. When this malicious auto-ware infects a network, it will act like a robot, mimic normal behavior of web access, and bypass the network firewall or intrusion detection system. Besides that, in a private and large network, with huge Hypertext Transfer Protocol (HTTP) traffic generated each day, communication behavior identification and classification of auto-ware is a challenge. In this paper, based on a previous study, analysis of auto-ware communication behavior, and with the addition of new features, a method for classification of HTTP auto-ware communication is proposed. For that, a Not Only Structured Query Language (NoSQL) database is applied to handle large volumes of unstructured HTTP requests captured every day. The method is tested with real HTTP traffic data collected through a proxy server of a private network, providing good results in the classification and detection of suspicious auto-ware web access.

***Keywords*:** Automated software, HTTP traffic analysis, Malware detection, NoSQL database

1. Introduction

Automated Hypertext Transfer Protocol (HTTP) software (auto-ware) could be classified into categories as follows [1]: normal software, such as anti-virus updaters; mail clients; browser toolbars; greyware, encompassing adware, spyware, and joke programs; malicious software acting as an HTTP-based botnet; and Trojan horses. Normal auto-ware can be controlled and is beneficial to the user; however, since HTTP greyware or malicious software penetrate the users' networks, they turn out to be internal threats. With it, attackers can conduct various types of application layer attacks, which are difficult to detect and/or prevent, such as denial of service (DoS), distributed DoS, and malware distribution.

In a private network, due to security threats, all direct Transmission Control Protocol/Internet Protocol (TCP/IP) outbound/inbound connections should be banned. But HTTP is an exception since, nowadays, application development is transferred more and more onto the web, and everything users need is found through web services. Therefore, in some ways, if HTTP auto-ware can infect a user's PC or network, it might still transparently communicate with its servers/sites. Furthermore, in a large private network, classification between types of HTTP auto-ware traffic is becoming tougher when huge numbers of requests are generated each day. To maintain communication, perform updates, or receive commands, all kinds of HTTP-based auto-ware have common characteristics in that they repetitively generate legal traffic and requests to their servers/domains. However, in the details, there are some sophisticated differences in the communications behavior of auto-ware with their sites.

- Malicious HTTP-based bots always follow the PULL method, where they connect to their command-and-control (C&C) server periodically in order to get commands and updates. The number of requests from malicious bots is not as high as normal auto-ware [2]

(e.g. updaters and downloaders), which simply generate requests with longer intervals than unusual malicious bots.

- A difference with malicious bots is that they often connect to one control domain and to a specific server resource, and undesirable applications or greyware, such as annoying adware or spyware, often report back to, or request new information from, the external master [8]. Therefore, they keep requests to their numerous advertising sites or uniform resource identifiers to update pop-ups or advertisements and commercial content areas.
- On the other hand, with auto-ware, there are no intervals or periodic patterns in users' web accesses. However, in recent years, many sites (e.g. online shopping malls, social media webpages) append advertisement paths to their sites and use JavaScript or Flash as an auto-aware process to automatically collect the advertising content as adware or spyware. Therefore, parts of a user's accessed sites can act as auto-ware.

In this paper, by using a Not Only Structured Query Language (NoSQL) database to handle large numbers of HTTP requests, a method for classification of HTTP auto-ware at the network level is proposed. In it, because of power and convenience in dealing with unstructured Extensible Markup Language (XML)/JavaScript Object Notation data, the fast-developing and easy-for-later-development MarkLogic NoSQL database [4] and XQuery [5] are suggested for the experiment. The method is tested with real traffic data generated from a university network, and good results are achieved in the classification and detection of malicious auto-ware web access.

The rest of this paper is organized as follows. The next section describes related works. Section 3 is dedicated to the description of the framework and methodology, including feature extraction. Phases of the proposed method are presented in Section 4. Experiment results and conclusions are expressed in sections 5 and 6, respectively.

2. Related Work

There is a considerable amount of research on HTTP malware detection. Ashley [6] suggested a method for detecting potential HTTP C&C activity based on repeated HTTP connections to a website. According to this, an algorithm was proposed for detecting HTTP polling activity. Wei Lu et al. [3] proposed a hierarchical framework using signature-based techniques to automatically discover malicious bots in a large-scale Wi-Fi Internet Service Provider network in which traffic is classified into different application communities by using payload signatures. These signatures were used to separate known network traffic from unknown traffic in order to decrease the false alarm rate. Eslahi et al. [2] proposed an approach to reduce HTTP botnet detection false alarms. In this research, high access–rate traffic, which might be other security threats, is filtered out. Other research [2, 6, 7] focused on botnet communication to a C&C server, but actually, HTTP threats come not just from malicious bots

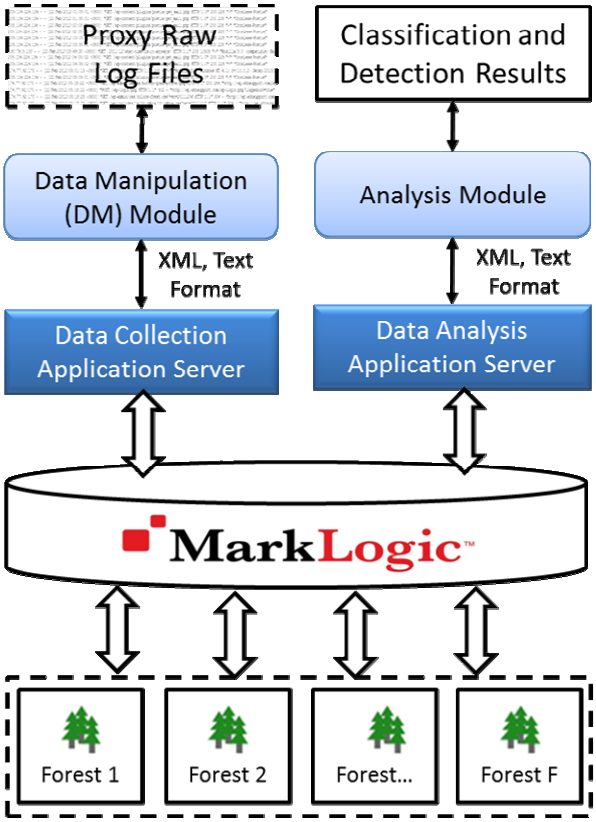

Fig. 1. General design of the observation framework using a MarkLogic database.

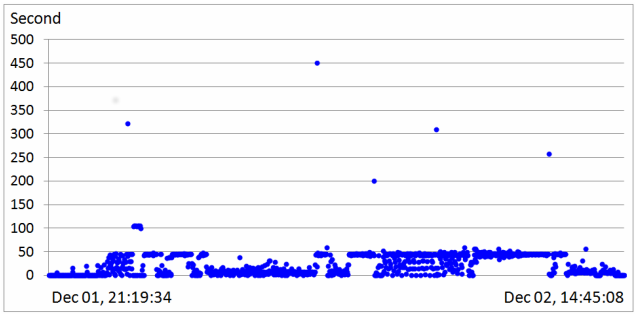

Fig. 2. Access graph of a client request to a URL.

but also from other types of software, such as key loggers or spyware, adware, or unauthorized peer-to-peer applications [8].

Bartlett et al. [8] proposed an approach to identifying low-rate periodic network traffic and changes in regular communication by auto-ware. Their research also focused on some types of auto-ware and monitored TCP flows to detect them, but in our paper, the target is not just detecting general types of auto-ware but also on particular uniform resource locators (URLs) where auto-ware requests were sent. In addition, our proposed method just collects and processes related HTTP traffic at the application layer, which helps reduce expensive processing, compared with processing TCP packets.

3. Framework and Methodology

In this paper, passive behavior is analyzed from collected HTTP requests in a certain network in order to extract features for the proposed method.

3.1 Implementation Framework

Data for observation and analysis were collected from the web proxy of a certain network that served about 2000 clients. Collected data were divided by day and saved into log files as raw data, from which a framework with two modules was implemented based on the MarkLogic NoSQL database, as shown in Fig. 1. Thereby, a MarkLogic cluster was set, and owing to optimizing performance in queries to the database, two XDBC application servers that include data collection and analysis along with a number of forests were configured. Therefore, development modules could work concurrently, as follows.

- The Data Manipulation Module reads raw log files and converts them to XML and text format before storage in the MarkLogic database through the data collection application.
- Core functions of the proposed method are implemented in the Analysis Module. It processes HTTP traffic and works with the database through a data analysis application server, after which it provides the result.

3.2 Feature Extraction

Based on the captured HTTP traffic, numerous extracted features are included.

- Client IP: Source IP address of a machine in the network that generated requests.
- Request Method: main methods of HTTP requests, i.e. POST/GET.
- Request date/time: Date and time when a client sends a request.
- URL: URL requested by a client IP but without parameters. Some normal servers are hacked, and some resource paths are exploited as C&C servers. Therefore, a non-parameter URL is used instead of a domain to help with more detail in the classification of auto-ware access behavior.
- Unique URL: Set of unique URLs requested by a client.
- Request Duration: Durations from two consecutive requests to the same URL.
- Request Number: Number of requests to a URL from a client during a period in the observation data.

3.3 Access Graph

With a client IP, 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 $R=\{r_1,r_2,...,r_N\}$ is a set of requests from a client to a server/webpage, and all r_i have the same webpage/server URL, as described in Fig. 2, then access graph G is a sequence including N-1 items, $G=\{g_1,g_2,...,g_{N-1}\}$, where g_i is a pair (t_i, d_i), in which t_i is the timing of request r_{i+1}, and d_i is the request interval between r_i and r_{i+1}. An example of access graph is shown 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 auto-ware 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 auto-ware and the webpage or server URLs.

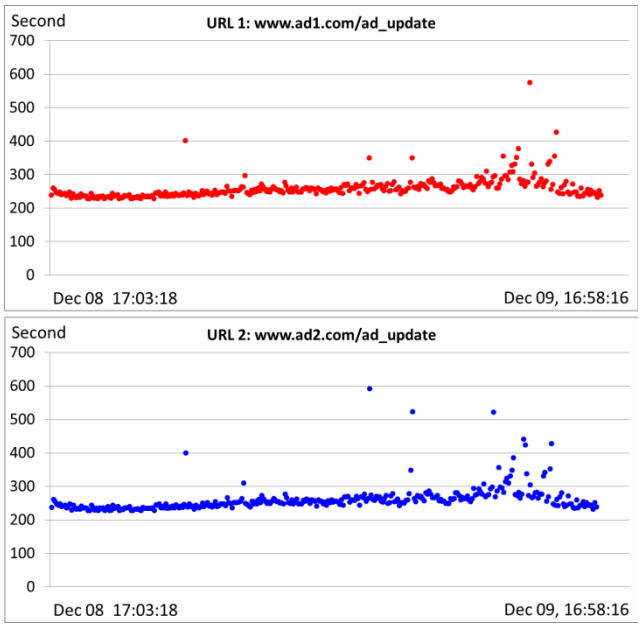

Fig. 3. Similar access graphs from a client IP to two different URLs.

Fig. 4. A malicious bot access graph.

By observation of captured data, even if they are different, if URLs are requested by the same auto-ware, then the access graphs look similar, as can be seen in Fig. 3. From this, to score the similarity in access behavior to URLs from a client, a distance value between access graphs is proposed and explained in the next subsection.

3.4 Access Graph Distance

Assume there are two access graphs: $A=(a_1,a_2,...,a_N)$ and $B=(b_1,b_2,...,b_M)$. Define the distance between two points a_i and b_j, which is calculated as a Euclidean distance

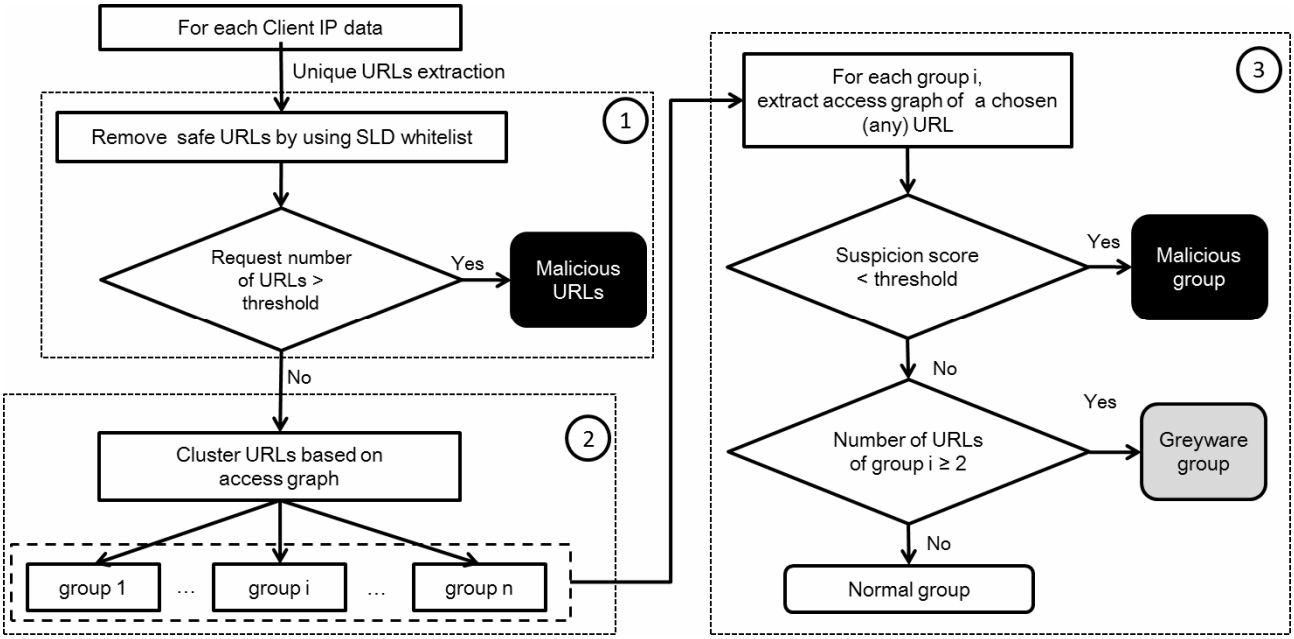

Fig. 5. Proposed method diagram implemented in the analysis module described in Fig. 1.

$d(a_i, b_j) = \| a_i - b_j \|$. From that, the distance between point a_i and graph B is defined as $d(a_i,B) = min \| a_i - b_j \|$ where $b_j \in B$. A generalized Hausdorff distance [5] for A and B is defined as

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

Based on Eq. (1), the modified Hausdorff (MH) distance [5] between graph A and B is

$$MHD(A,B) = max(d(A,B), d(B,A)) \quad (2)$$

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

3.5 Suspicion Score

As described in earlier research [1, 2], malicious bots connect to their command and control server periodically in order to get commands and updates; therefore, there is almost no large variation in the access graph from a malicious bot to its C&C, as can be seen in Fig. 4. Assuming that a bot access graph is specified and denoted as $X=(x_1,...x_N)$, a suspicion score will be defined as a *coefficient of variation* of X. A smaller suspicion score shows that a URL is more suspicion:

$$SuspicionScore(X) = \frac{\sigma}{\mu} \quad (3)$$

where σ and μ, respectively, are the standard deviation and the mean of X.

4. Proposed Method

Based on the auto-ware communication behavior described in Section 1 and the observation of access graphs in Section 3, a classification and detection method, including three phases, is proposed as illustrated in Fig. 5, with details below.

The first phase is pre-processing. For each client IP, the one-day HTTP traffic features are extracted and pre-processed. To remove safe URLs, two methods are applied: the first is to filter URL requests from the client IP through a white list of second-level domains (SLDs). This filter method is described by Chen et al. [9], and according to them, the tokens in the URLs of phishing websites are less consistent with the website content, when compared with legitimate websites. Therefore, a domain name that contains an SLD defined in the SLD white list is marked as benign. The second method is based on the number of requests to a URL from a client IP. Based on the observed number of requests from auto-ware to a URL, it can be seen that auto-ware accesses a URL many times within a specific time duration. Therefore, if the number of requests to a URL is too small, they seem to not be requests by auto-ware. In this paper, a threshold equal to 20 is set to filter out this type of very low–access URL. Conversely, URLs that are accessed extremely quickly in a short time suggest malicious auto-ware access.

Group clustering is the second phase, after the pre-processing phase, where the remaining URLs will be clustered into a number of groups. A simple approach to decide the group for any two URLs is to calculate the MH distance between the two access graphs of the URLs, and if this MH distance is smaller than a threshold, they will be in the same graph. However, this method consumes processing time. To overcome this issue, in our method, clustering is proposed, as shown in Fig. 6, in which a seed vector (S vector) is chosen constantly, and di and dj, respectively, are

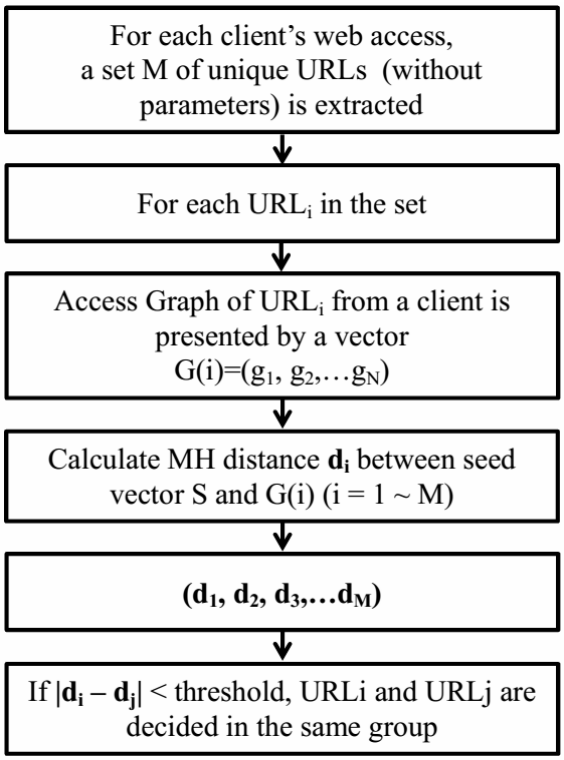

Fig. 6. Clustering phase sequence.

MH distances between the access graph of URLi and the access graph of URLj to the S vector. URLi and URLj are placed in the same group if |di - dj| is not greater than a threshold, which in this implementation is 0.006.

The third phase is classification. For each group, one URL (any of the ones in the group) is chosen, the access graph of this URL is extracted, and the suspicion score (as explained in Section 3.4) is calculated to detect if it is malicious or not by comparing it to a threshold, which is proposed as 0.04. Remaining groups will be detected by counting the number of unique URLs in the group. As analyzed in Section 1, different from malicious bots, greyware commonly accesses various URLs. Therefore, a group having number of unique URLs not less than 2 will be marked as a greyware group.

5. Experiment Results

For experiment purposes, 11 client IPs' HTTP traffic from a university network were analyzed and classified through the proposed method, and experiment data are summarized in Table 1. Two Zeus [10] bots are installed in a client with difference intervals in communication for C&C. The total number of requests was 1,372,056.

After the pre-processing phase, as described in Section 4 and Fig. 5, a set of unique URLs was established with 1000 URLs, as seen in Table 1. In that set, there are two URLs that were accessed at extremely high speed by IP1 and IP9, e.g. IP1 accessed one URL 237,291 times in about 122 minutes (7329 seconds, or nearly 32.4 requests per second). By checking manually, these URLs were from domains that contain unwanted software and are marked as malicious by

Table 1. Experiment Data.

IP	Requests (with Parameters)	Unique URLs (without Parameters)	After Pre-processing (Phase 1)
IP1	238,018	201	1
IP2	70,645	4,356	121
IP3	67,597	15,734	108
IP4	161,595	1,462	125
IP5	58,874	1,014	101
IP6	64,701	613	107
IP7	119,376	3,458	214
IP8	96,594	4,596	67
IP9	318,609	2,338	3
IP10	153,742	4,915	138
IP11	22,305	4,855	15
Total	**1,372,056**	**43,542**	**1,000**

Table 2. Experiment Results.

Client IP	Group clustering (Phase 2)		Classification (Phase 3)				
			Greyware		Malicious		Normal (Groups 1 URL)
	Group	URL	Group	URL	Group	URL	URL
IP1	1	1			1	1	
IP2	25	121	7	103			18
IP3	56	108	20	73			36
IP4	13	125	4	120			5
IP5	14	101	6	93			8
IP6	15	107	3	88			12
IP7	53	214	22	183			31
IP8	36	67	14	45			22
IP9	3	3			1	1	2
IP10	60	138	28	105			33
IP11	14	15	1	2	2	2	10
Total	**290**	**1000**	**105**	**812**	**4**	**4**	**177**

many network security companies and software. The remaining 998 URLs were classified into 290 groups by running them through the classification phase.

As seen in the results summarized in Table 2, two URLs requested from IP11 were detected as malicious, and they were matched with C&C servers communicated with by the installed Zeus bots. All the detected greyware groups were confirmed as coming from shopping sites, news and social media, and advertising companies. All normal groups detected were groups with only one URL; however, in this class, there are actually 159 domains belonging to greyware groups, or that were unknown but detected as normal. This gave a 15.9% false negative rate, and the accuracy rate reached 84.1%.

6. Conclusions and Future Work

In this paper, a new method is proposed to detect and classify auto-ware communication behavior based on

HTTP traffic. A framework using a NoSQL database is implemented to help deal with huge amounts of captured traffic. The convenience of MarkLogic to manipulate unstructured data is really helpful in reducing programming time. In addition, providing XQuery in MarkLogic is a powerful way to extract needed information for data analytics or analysis.

There are some reasons contributing to the false negative rate. First, even auto-ware commonly communicates with sites with the same behavior; however, some rare cases of auto-ware requests are different. Second, in this method, a constant seed vector is chosen to evaluate the similarity between access graphs, and this can affect the error rate. Therefore, a better vector-choice method needs to be considered in the next step. Based on this result, with an objective to make the framework more powerful, development of the current method into a combination of MarkLogic and Hadoop, which is known as a great tool for distributed storage and processing, is planned for future work.

References

[1] Manh Cong Tran and Yasuhiro Nakamura, "In-Host Communication Pattern Observed for Suspicious HTTP-Based Auto-Ware Detection," International Journal of Computer and Communication Engineering, vol4, no. 6, pp. 379-389, 2015. Article (CrossRef Link).

[2] Meisam Eslahi, Habibah Hashim and Noorita Tahir, "An Efficient False Alarm Reduction Approach in HTTP-based Botnet Detection," in Proc. IEEE Symposium on Computers & Informatics, pp. 201-205, April 2013. Article (CrossRef Link).

[3] MarkLogic database, Article (CrossRef Link), last visit: October, 2015.

[4] MarkLogic Development Document, Article (CrossRef Link), last visit: October, 2015.

[5] Dubuisson, M.-P.; Jain, A.K., "A modified Hausdorff distance for object matching," in Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on , vol.1, pp.566-568 , 9-13 Oct 1994. Article (CrossRef Link)

[6] Daryl Ashley, "An algorithm for http bot detection," University of Texas at Austin - Information Security Office, 2011.

[7] Wei Lu, Mahbod Tavallaee, and Ali Akbar Ghorbani, "Automatic discovery of botnet communities on large-scale communication networks," in Proc. the 4th International Symposium on Information, Computer, and Communications Security, Sydney: Australia, pp. 1-10, 2009. Article (CrossRef Link)

[8] Bartlett, G.; Heidemann, J.; Papadopoulos, C., "Low-rate, flow-level periodicity detection," in Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on, pp.804-809, April 2011. Article (CrossRef Link)

[9] Yi-Shin Chen; Yi-Hsuan Yu; Huei-Sin Liu; Pang-Chieh Wang, "Detect phishing by checking content consistency," in Information Reuse and Integration (IRI), 2014 IEEE 15th International Conference on, pp.109-119, Aug. 2014. Article (CrossRef Link).

[10] Zeus: King of Bots, "Article (CrossRef Link)", last visit: September, 2015.

Manh Cong Tran was born in Nam Dinh, Vietnam, in 1981. He graduated with a bachelor's degree and his master's degree in computer science from Le Quy Don Technical University of Vietnam in 2004 and 2007, respectively. He is currently a PhD candidate in the Department of Computer Science, National Defense Academy, Yokosuka, Kanagawa, Japan. His current research interests include network traffic classification/analysis and anomalous/ malicious detection. He is also employed in Le Quy Don Technical University, Hanoi, Vietnam.

Yasuhiro Nakamura was born in Tokyo, Japan, in 1959. He graduated from the National Defense Academy (NDA), Japan, in 1982, and received a PhD in engineering from Keio University, Tokyo, Japan, in 1990. He was a Major in Japan's ground self-defense force until 1995, and has been a Professor in the Department of Computer Science, NDA, since 2011. He has been the Chairperson of the Department of Computer Science since 2013. His current research interests are in computer network security and network system management in Yokosuka, Kanagawa, Japan. Dr. Nakamura is a member of The Institute of Electronics, Information and Communication Engineers of Japan (IEICE), the Information Processing Society of Japan (IPSJ) and the Institute of Image Electronics Engineers of Japan (IIEEJ).