HAS-Analyzer: Detecting HTTP-based C&C based on the Analysis of HTTP Activity Sets

Sung-Jin Kim, Sungryoul Lee and Byungchul Bae
Attached Institute of Electronics and Telecommunications Research Institute (ETRI),
Yuseong P.O.Box1, Daejeon, Korea
[e-mail: {ksj1230,srlee0525,bcbae}@ensec.re.kr]
*Corresponding author: Sungryoul Lee

Received December 17, 2013; revised February 19, 2014; revised March 18, 2014; accepted April 12, 2014; published May 29, 2014

Abstract

Because HTTP-related ports are allowed through firewalls, they are an obvious point for launching cyber attacks. In particular, malware uses HTTP protocols to communicate with their master servers. We call this an HTTP-based command and control (C&C) server. Most previous studies concentrated on the behavioral pattern of C&Cs. However, these approaches need a well-defined white list to reduce the false positive rate because there are many benign applications, such as automatic update checks and web refreshes, that have a periodic access pattern. In this paper, we focus on finding new discriminative features of HTTP-based C&Cs by analyzing HTTP activity sets. First, a C&C shows a few connections at a time (low density). Second, the content of a request or a response is changed frequently among consecutive C&Cs (high content variability). Based on these two features, we propose a novel C&C analysis mechanism that detects the HTTP-based C&C. The HAS-Analyzer can classify the HTTP-based C&C with an accuracy of more than 96% and a false positive rate of 1.3% without using any white list.

Keywords: Network security, botnet detection, HTTP-based C&C

A preliminary version of this paper appeared in ACSAC 2012, Deceber 3-7, Orlando, Florida, USA. This version includes a concrete analysis and supporting implementation results on HAS-Analyzer.

http://dx.doi.org/10.3837/tiis.2014.05.017

1. Introduction

Recently, HTTP-based malware has become more prevalent and has inflicted tremendous damage on many industries and government organizations through DDoS Attack, spamming, etc. Moreover, a number of web exploit kits using zero-day exploits [9] makes it easy to deploy aware to a broad range of victims. Arguably, the most crucial attack of malware is data theft. This type of malware typically exfiltrates the harvested data from infected machines to the corresponding C&C server. It might be developed into a targeted Advanced Persistent Threat (APT). Thus, the research community has developed a number of behavioral pattern-based techniques for distinguishing HTTP-based C&Cs from normal web accesses on benign web servers. Most previous research has focused on the regular access pattern of C&Cs, such as BotSniffer [8], BOTFINDER [20] and DISCLOSURE [1]. They concentrate on a periodic access pattern, response crowd, flow sizes, etc. However, these approaches need a well-defined whitelist to reduce false positives because there are many benign applications (e.g., automatic updatechecks, web refreshes, etc.) that have a regular access pattern.

In this paper, we try to find the discriminative features for distinguishing C&C flows based on HTTP activity set which is a set of request-response pairs generated by the initial application request for a URI. By coupling multiple web connections into HTTP activity sets, we could derive two discriminative feature sets.

- Low density: A C&C channel utilizes a small portion of network resources. The corresponding features are the average referer tree depth, the average number of different destination domains, the average number of requests and the average number of bytes of exchanged data.
- High content variability: The content in a series of C&C connections changes frequently at each time during the process of stealing data or updatingcommands. The corresponding features are the minimum and average similarity scores of the content, and the number of significant changes (formally defined later).

Based on these observations, we present the HAS-Analyzer, a system that detects malware-infected machines and the corresponding C&C servers by using only network analysis. The HAS-Analyzer leverages the two feature sets, low density and high content variability. This means that HTTP-based malware exchanges different data to the C&C server at each connection with a small amount of traffic.

We demonstrated the effectiveness of the two feature sets. Our evaluation is based on 126,018 active C&C connections from 500 known malware samples and 1.2 TB of a benign dataset. Though we did not use any white list or reputation system, we were able to archive a high accuracy and low false positive rate by using the HAS-Analyzer.

Main Contribution. The primary contribution of this paper is the introduction of two discriminative power feature sets of HTTP-based C&Cs. We first present a new data structure, the HTTP activity set, coupled with multiple web connections. Based on the analysis of the HTTP activity set, we derived two unique feature sets for HTTP- based C&Cs: low density and high content variability. We also present the HAS-analyzer, which detects HTTP-based C&Cs with a high accuracy and a low false positive rate without using any reputation systems. We evaluated the accuracy of the HAS-analyzer against HTTP-based C&C experiences

through a virtual machine experimentation platform using more than 500 recently captured HTTP-based malware examples. We validate our proposed system by demonstrating how our detection engine successfully identifies the HTTP-based C&C on normal web connections.

2. Background

2.1 Case Study of HTTP-based C&C

In traditional HTTP-based botnets, the attacker simply sets the command in the data at a C&C server. Then the botnets frequently connect to their C&C servers with a predefined delay. Today, botnets not only just receive a command but also have the ability to harvest personal data from the compromised machine and exfiltrate them to the corresponding C&C server. **Fig. 1** shows the HTTP-based C&C behavior. They typically split the harvested data into small chunks and then exfiltrate them to the predefined URL using a GET or POST method. For updating commands, botnets insert current status information into the reporting request at each connection. There are several well-known HTTP-based botnets, for example, Zeus [26], which is designed mainly to steal financial data. The bots of this botnet periodically connect to the C&C server with a URL such as http://.../gate.php.

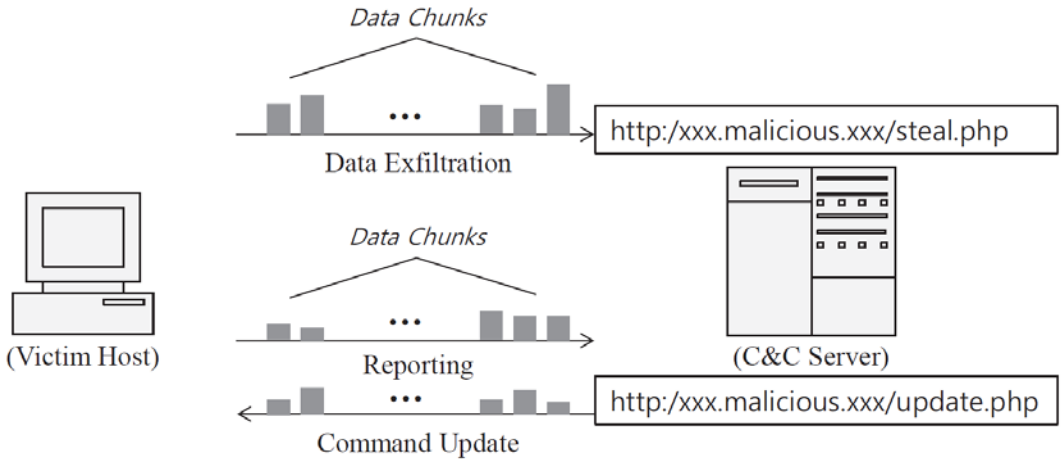

Fig. 1. Behaviors of HTTP-based C&C

2.2 Motivation

As depicted in **Fig. 1**, most HTTP-based C&C traffic is low volume. This is a strategy of C&C to make their detection difficult, especially if there is only a small number of C&C connections in the monitored traffic. We take this strategy as the first feature of HTTP-based C&C. Thus, a small number of connections or a small number of total transferred bytes from a host may be an indicator for C&C behavior. In addition, the exfiltrated data changes on each connection as the harvested data from a victim host change. In addition, the command or current status of a victim host changes at each connection. Thus, a series of attached content in C&C connections will change from time to time. We prefer to denote such phenomena as high content variability. To check high content variability, we compare the similarities in attached content of C&C

connections heading for the same URI. Before illustrating the new features in detail, in the next section we outline a new data structure to dissect multiple web connections into an activity set.

3. HTTP Activity Set (HAS)

3.1 Definition

In order to extract discriminative features, we first define an HTTP activity set as a new data structure. An HTTP activity set is a correlated set of request-response pairs generated by the initial application request for an external web server. We refer to it as a 'HAS' in the rest of this paper. Specifically, a HAS can be generated by human-driven applications (e.g., web browser) or by automatic programs (e.g., RSS feeds, web refresh or bots).

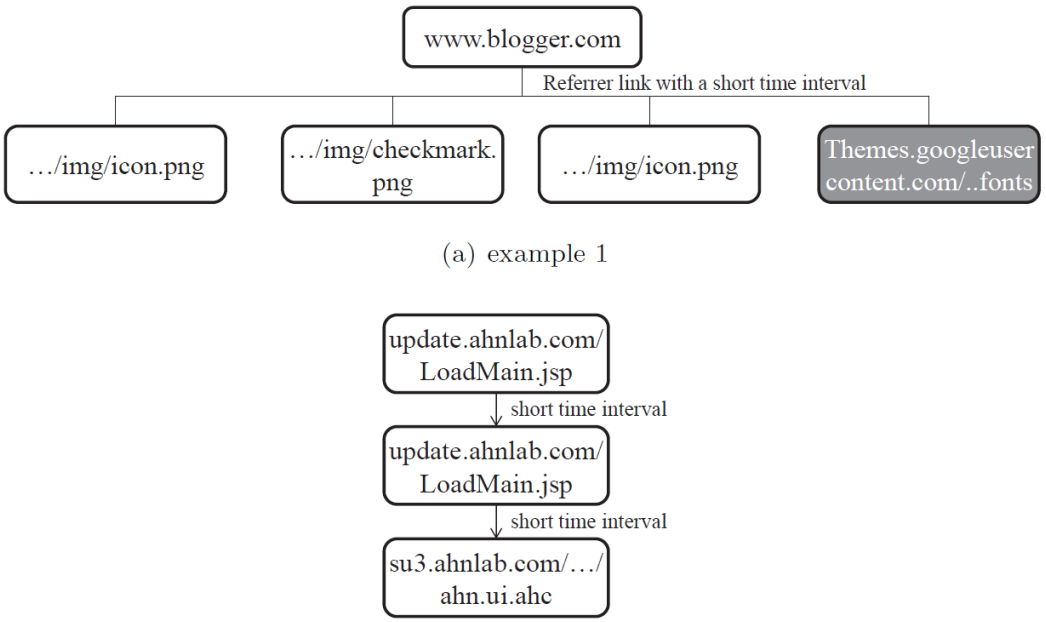

Fig. 2. Examples of HAS

Fig. 2 shows two examples of the HAS. In both cases, we can see that coupled web connections may have different destination addresses. For example, the domain "themes.googleusercontent.com" and "www.blogger.com" are correlated by a referer link, depicted as example 1 (a) in **Fig. 2** and "su3.ahnlab.com" and "update.ahnlab.com" are also correlated by the request time interval depicted as example 2(b) in **Fig. 2**. We will use both the referer information and the request time interval to generate HTTP activity sets.

3.2 Metadata Structure

In order to couple separate web connections into one HAS, it is essential to develop a custom data structure that provides access to the syntactic fields of the web traffic. First, we

reassemble the TCP stream and parse the HTTP traffic. Then, we design a record of the HASs that have a same source IP. We extract the time stamp of request time (*ts*), destination domain (*domain*), URL path (*url*), referer information (*ref*), and useragent (*ua*) as well as the content in the request and response (*Contentreq*, *Contentresp*). Finally, we define a data structure *R* as follows.

$$R := \{ts, domain, url, ref, ua, Contentreq, Contentresp\} \quad (1)$$

A HAS is composed of a set of *R* and has a unique identifier that is a full request URI (*domain* + *uri*) in the first *R* of the HAS.

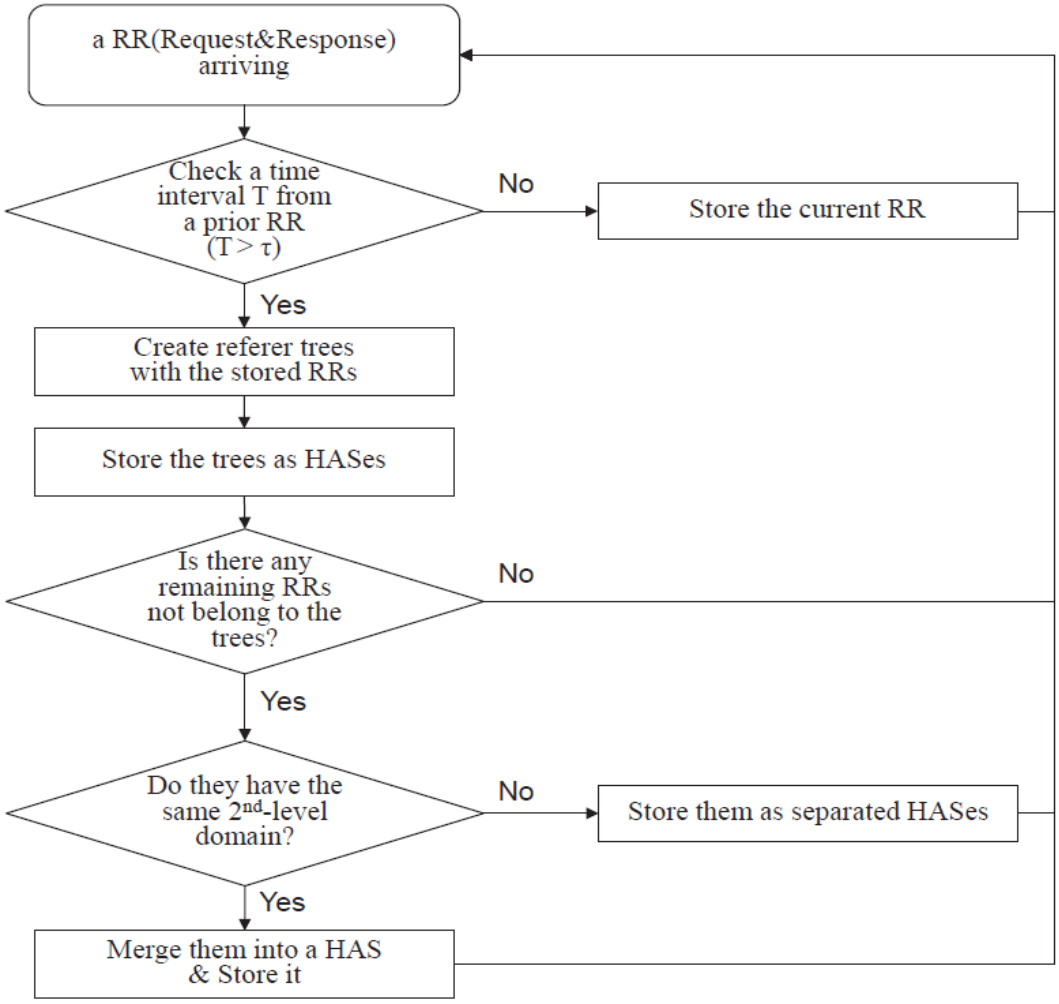

Fig. 3. Flowchart of method to extract HAS

3.3 Methodology

Now, we introduce a method to couple the separate web connections into HAS using network traffic analysis. There are three main factors as follows.

- Request time interval: Since an operation of web applications generates a set of web connections within a short time interval, we regard these web connections as a group.
- Referer information: Using the referer information, we construct a referer tree, where the nodes are the downloaded content, *c1* and *c2*, and the edge from node c1 to node *c2* means that an application followed a link from *c1* to *c2*. Note that the root of a referer tree represents an initial request by an application without following a link. In most cases, a referer tree is generated by webbrowsers and it could be regarded as an activity set as depicted in (a) example 1 in **Fig. 2**.
- Second-level domain: Automatic programs usually do not record referer information into web requests. Thus, we need other information to group web connections of automatic programs. We can simply infer that second-level domains could be a clue to group those connections. Depicted as (b) example 2 in **Fig. 2**, the antivirus program commonly communicates with its web server, anhlab.com. In this case, ahnlab is the second-level domain of .com TLD (Top-level domain).

The flowchart described in **Fig. 3** explains how HASs are extracted by using the above three factors. Finally, we obtained the two cases of HAS as depicted in **Fig. 2**. One is a constructed referer tree within a short time interval ((a) example 1). Another is a set containing the same second-level domain within a short time interval ((b) example 2). The time interval should be lower than the threshold τ. We empirically determined the value τ as 1.8 (sec) by measuring the maximum time intervals in our extensive experiments.

4. Feature Selection

We introduce several discriminative features based on analysis of the HAS. These features are derived from low density and high content variability for the HAS.

4.1 Low Density

The first feature sets in our analysis mechanism are based on the density in a series of HASs. An HTTP-based C&C uses a small portion of network resources in order to guarantee its stealth because the user of the compromised machine would not notice them. Following this premise, the derived features are:

- The average referer tree depth: We can easily observe that the referer information of a C&C contains a spoofed value or no value. That means that the HAS of the C&C tends not to generate a referer tree, whereas a benign HAS can create a deep referer tree.
- The average number of different destination domains: For attackers, it is costly to maintain a number of C&C servers at the same time. That means that the malware programs typically communicate with one C&C server during a given period of time.
- The average number of requests: In order to maintain the stealth of the C&C connection, the malware often uses a small number of requests each time.
- The average bytes of exchanged data: The long and steady process of data exfiltration can reveal the characteristics of data theft malware. Data theft malware often splits harvested data into small pieces and steadily sends them to the corresponding C&C server. Thus, the average bytes of exchanged data are expected to have limited value.

4.2 High Content variability

To extract the second feature set, we inspect the content carried in each request and response. In our observation, the exchanged content for the same URL are changed significantly during the process of data exfiltration or updating commands. The properties of the content specify the bodies of the request, response, parameters in the URL path, and the additional header values in the HTTP request and response. There are several methods for checking the similarity between two sets of content. In our work, we aim to pick up C&C messages on normal web contents that have a form of plain or encoded ASCII texts, or pre-defined binary data form. Due to the diversity of content format used in malware, we require the standard fingerprint which makes it applicable for different type of content. Thus, we adopt SimHash [18] as our similarity function where it measures the bitwise hamming distances between hash values. SimHash specifies the distribution on a family of hash functions $H = \{h\}$ such that for two objects x_i and x_j,

$$PR_{h \in H}\{h(x_i) = h(x_j)\} = 1 - \frac{\theta(x_i, x_j)}{x} \tag{2}$$

where $\theta(x_i, x_j)$ is the angle between x_i and x_j . Another reason for choosing SimHash is the privacy concern. When we parse the HTTP payload of a web connection, we store the hash value, x_i , for content i instead of storing raw content directly without sanitization. This issue is further discussed in section 8.

- The minimum and average similarity score: The C&C server tends to receive and return different content for the same request URI from clients in contrast to a benign web server. We calculate the similarity scores of each property in the content. We then select the minimum value among them. The score has a value between 0 and 1. A low score means that the variability is high at a certain point due to the data exfiltration. We also check the average similarity score.
- The number of significant changes: We check how many significant changes (e.g., similarity score < 0.5) occurred in a given number of observations. This is normalized as n/l, where n is the number of significant changes and l is the given number of observation. The score has a value between 0 and 1. A high score means that the variability is high in a series of web connections.

5. Proposed System

5.1 Overview

Our proposed system, the HAS-Analyzer, is an HTTP-based C&C detection system designed to identify both infected hosts and C&C servers. **Fig. 4** shows an overview of the system architecture. The HAS-Analyzer is composed of three main phases: HAS extraction, feature extraction and training-detection. Our method for extracting a HAS is introduced in the last part of this section. The feature extraction is carried out in two stages that contain a density check and content variability check based on our feature selection. We can finally pick out a suspicious HAS by putting feature vectors into the classifier.

HTTP Activity Set (HAS) Extraction. A HAS consists of multiple pairs of records. Each record contains some values in the HTTP request header and bodies of both the request and the response. The criteria for classifying the HAS is the referer information and the request time interval. HAS extraction finally generates a series of HASs that have the same IP address and the same first record. The detailed method for extracting a HAS was introduced in the Section 3.

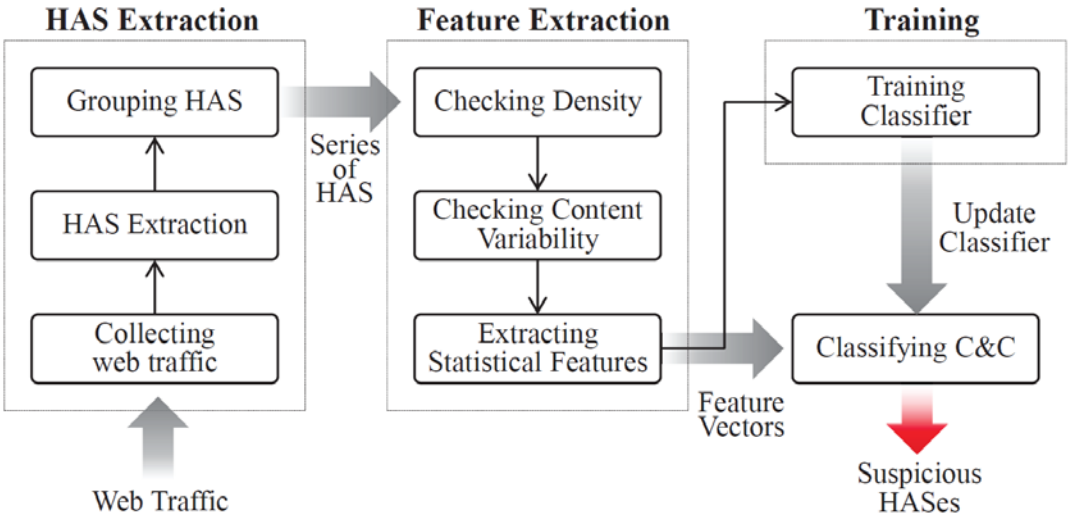

Fig. 4. Overview of proposed analysis mechanism

Feature Extraction. The feature extraction has two sub-components: checking the density of the HAS (HAS-D) and checking the content variability of the HAS (HAS-C). These components wait until a series of HASs is generated from the HAS extraction. When more than n HASs are collected (in our implementation, n=20), the HAS-Analyzer checks the density and content variability of the HAS. The HAS-D component checks the average referer tree depth, the average number of different destination domains, the average number of requests and the average number of bytes of the exchanged data. Next, the HAS-C component checks the minimum similarity score and the number of signifficant changes. However, it may be difficult to compute all the similarity scores of the content between two adjacent HASs. Since some benign HASs contain hundreds of records (e.g., visiting web portal sites), we compute the similarity scores of the first w content that have the same request URI. We determined the value w as 5 because the maximum number of requests in the C&C HAS was 5 in our dataset.

Training. We trained the HAS-Analyzer with 121,529 C&C HASs and 1.6 TB of benign web traffic consisting of 683,223 benign HASs. We split these datasets into atraining set (70% of dataset) and a test set (the remaining 30%). Then we created a detection model by using a WEKA machine learning toolbox with 10-fold cross validation. We used three machine learning algorithms for the classifier, including Naïve Bayes, J48 decision tree [17] and Random Forest [11]. We compare the accuracy for these algorithms in the evaluation.

5.2 Details of HAS Extraction and Feature Extraction

To assist in understanding this system, we texplain additional information about HAS extraction and feature extraction. In **Fig. 5**, we give a description of how to extract a HAS and features through network traffic analysis. HAS extraction and the feature extraction consist of three steps, HAS-D1, HAS-D2 and HAS-CC. In HAS-D1, if the time interval between two adjacent requests exceeds the threshold τ , the HAS-Analyzer divides it into separate groups and then constructs referer trees in HAS-D2. As explained in the owchart of **Fig. 3**, generated referer trees become HASs and the rest of the RR (Request/Response) -pairs having the same second-level domain are merged into one HAS. The density features such as *r* (the number ofrequests) and *d* (referer tree depth) are calculated at this step. For example, at thetop of the description, four groups are generated since their request time interval is shorter than τ . Then the HAS 1-1 is generated from group 1 by constructing a referer tree, and the rest of RR-pairs that have the same second-level domain become HAS 1-2. Finally, in HAS-CC, the HAS-Analyzer groups a series of HASs that have the same identifier, such as the same initial web request. Therefore, HAS 1-2, HAS 2and HAS 4 are grouped into a series. The high content variability features are then extracted.

6. Evaluation

We performed a feature evaluation on our dataset. After the HAS-Analyzer was trained from a given training set, we evaluated the accuracy of the HAS-Analyzer for each feature over the test set. We used the 12 cores of Intel Xeon processor 2.66 Ghz with 16GB memories for the evaluation.

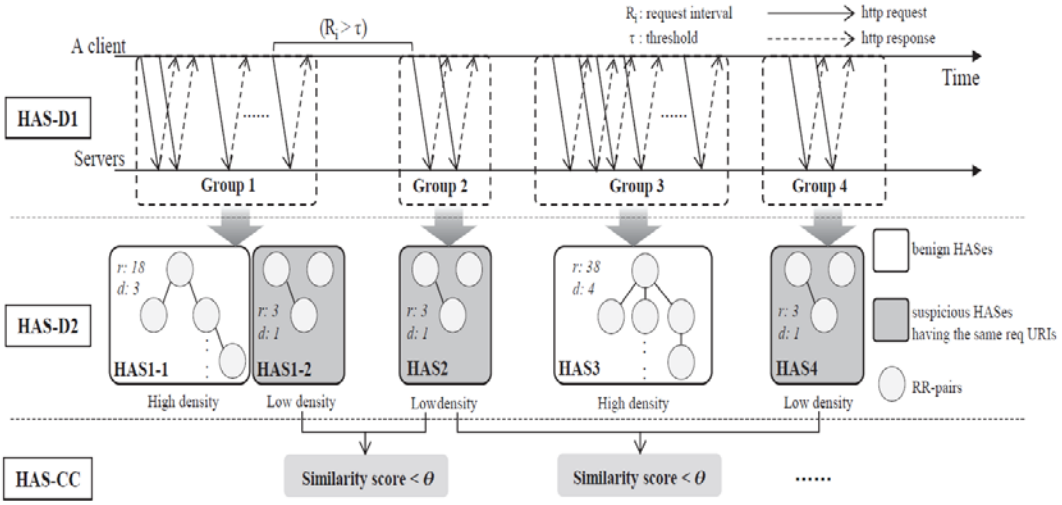

Fig. 5. Description of method to extract HAS and features

6.1 Dataset

Our dataset is based on about 500 malware samples and benign web traffic collected from an access network at our research institute. The Trace-M and Trace-B were used to train the HAS-Analyzer and the Trace-T was used to evaluate the HAS-Analyzer.

Trace-M (Malicious). Each C&C trace lasted for several hours in daytime. The Trace-M was collected from several sources. First, we collected HTTP-based C&Cs through implementing a VM-based system that was automatically connected to exploit URLs provided by the Malware Domain List [14] . The most common web exploit kits were the blackhole exploit kit and the cool exploit kit, and the most common type of malicious payload was Zeus bots. However, since the traces contain non-active C&C servers, we selected only active HTTP-based C&C traces. During the capturing, we obtained the banking trojans and some of info stealers. The other source wasVirusTotal [21]. We collected bot samples from VirusTotal. The set contained root kits, key loggers and backdoors used in APT campaigns. We executed the collected malware programs on a VMWare Workstation with Windows XP SP2 using TCPdump to trace the web traffic for an execution period of several hours. We also filtered out the non-active traffic, where the C&C server was dead. Finally, we collected 125,192 active C&C HASs. The data composition of Trace-M is depicted in **Table 1**.

Table 1. Full Description of Datasets

Datasets	Types	Examples
Trace-M	Banking Trojans (66.1%)	Zeus, Spyeye, Citadel, Tinba, Stabuniq
	Rootkits (12.6%)	Torpig, Xpaj
	APT Components (9.3%)	Taidoor, Ixeshe, Enfal, Murcy, win32.daws
Trace-B	Web Browsers (88.2%)	IE, Chrome, Safari, Mozila, Opera
	Non-Web Browsers (11.8%)	Alyac, V3, Twitter, Facebook, Dropbox, Simple RSS reader, iTunes, Others

Trace-B (Benign). Trace-B was captured on the access network in our laboratory. The total size of Trace-B was 1.2 TB and had full traces of web connections communicating with 3,879 benign web servers. The Trace-B consisted of web browsing traffic (88.2%) generated by employees and non-web browsing traffic (11.8%) generated by automatic programs. We sorted two groups by checking the destination server address and the user-agent field of the HTTP request. The non-web browsing traffic is important in our work because it is likely to cause false positives. It may generate false positives in the HAS-Analyzer when we use a low-density feature set. The false positives caused by non-web browsing traffic will be treated after the evaluation. The data composition of Trace-B is also depicted in **Table 1**.

Trace-T (Test). For testing, we overlaid traces from Trace-M instances onto our recorded laboratory network traffic and assigned the malware traffic to originate from randomly selected internal hosts observed to be active during that hour. This made our testing scenario much more realistic, because the internal hosts to be identified still exhibit their normal connection patterns, in addition to subtle C&C activities. In the Trace-T dataset, there were about 90 GB of benign web connections and 86,759 C&C connections..

6.2 Classifier Determination

To determine the best classifiers in the HAS-Analyzer, we evaluated the three standard classifiers in terms of accuracy (ACC) and false positive rate (FPR). **Table 2** shows the performance evaluation of the classifiers over the training set across the two proposed feature sets. As shown in **Table 2**, training on all the features suggests that Radom Forest [11] has the best performance. Therefore, we chose Random Forest as our classifier.

Table 2. Performance Analysis of Three Classifiers

Classifier	ACC (%)	FPR (%)
Naïve Bases	84.1	5.5
J48 Decistion Tree	91.7	9.6
Random Forest	96.9	1.3

6.3 Feature Evaluation

We performed all our experiments on a set of a fraction in HTTP-based C&C traces on a Trace-T dataset. **Fig. 6** shows the accuracy (ACC), false positive rate (FPR) and false negative rate (FNR) for the evaluation. The features we evaluated were the following: the average referer tree depth (D_1), the average number of different destination domains (D_2), the average number of requests (D_3), the average bytes of exchanged data (D_4), the minimum similarity score (C_1), the average similarity score (C_2), the number of significant changes (C 3), the combination of all low density features (D all), the combination of all high content variability features (C_all), and finally, the combination of all features (D+C). The best performance was in D+C. In low density features, the average referer tree depth (D_1) and the number of different destination domains (D_2) had over 80% of accuracy with no false negatives, while the average number of requests (D_3) and the average bytes of exchanged data (D_4) had an accuracy of 70% with a low false negative rate (about 3%). And, in high content variability features, all the features had an accuracy of 90% with low false positive rate (lower than 1%) and a little false negative rate (about 1%).

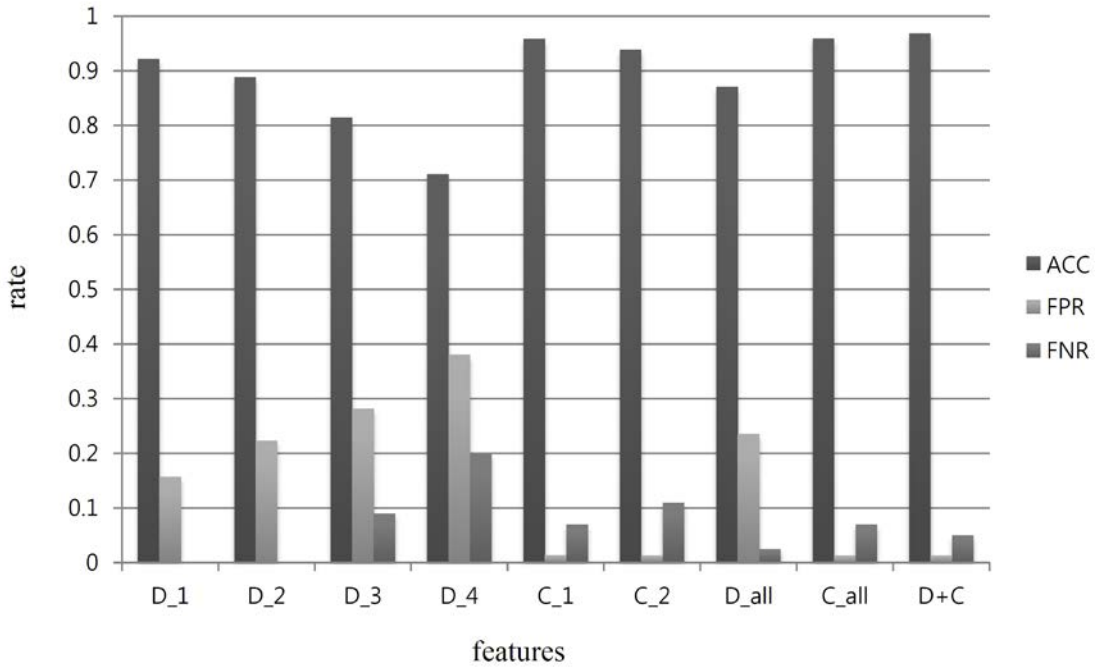

Fig. 6. Accuracy(ACC), false positives(FP), false negative(FN) for the different combination of features

Relationship between Two Feature Sets. As shown in **Fig. 6**, the high content variability feature set seemed to have a slightly better performance than the low density feature set. However, high content variability features had some false negatives instead of relatively low false positives. On the other hand, low density features had high false positives instead of relatively low false negatives. By comparing the results of D all, C all and D+C, we inferred the relationship between the two feature sets. As shown in **Table 3**, the low density feature set contributed to reducing false negatives by 1.95% and improving the overall accuracy slightly. Meanwhile, as shown in **Table 4**, the high content variability feature set contributed to reducing false positives by 22.23% and improving the overall accuracy by 9.82% with slightly increased false negatives. Thus, we concluded that the two feature sets are in the complementary relationship.

False Positives and False Negatives. We also analyzed the false positives and false negatives that occurred in our feature evaluation. The low density characteristic was often observed in benign automatic programs such as web refreshes, facebook, twitter, dropbox and many other automatic programs. These automatic HASs were regarded as C&C HASs that caused false positives. The high content variability characteristic can also be observed in benign HASs. Representative examples are dynamic web pages using server side scripting such as Active Server Page or Java Server Page. Suppose that a dynamic web page has content update scripts with pre-defined time intervals. As time goes by, the content of such a page will change continuously. Accordingly, the minimum similarity score or the average similarity score may be lower than the threshold θ. Then, false positives occasionally occur. False negatives occurred in the following situation. When malware generated simple heartbeat connections without updating commands or exfiltrating data, the content in the connections remained constant for a period of time. Such malware caused false negatives. As we discussed the relationship between the two feature sets, these issues can be mitigated by using the combination of all proposed features.

Table 3. Contribution of Low Density Feature Set on the Overall Accuracy

Measures	With D_all	With D+C	Change
ACC (%)	87.04	96.86	9.82(↑)
FPR (%)	23.52	1.29	22.23(↓)
FNR (%)	2.41	4.98	2.57(↓)

Table 4. Contribution of High Content Variability Feature Set on the Overall Accuracy

Measures	With C_all	With D+C	Change
ACC (%)	95.88	96.86	0.98(↑)
FPR (%)	1.28	1.29	0.1(↑)
FNR (%)	6.93	4.98	1.95(↓)

7. Related Work

Regular Access Pattern Analysis Approach. The regular traffic pattern is a strong signal in C&C communication. Thus, many previous studies presented the premise that C&C activity shows regular access patterns [1, 6, 8, 10, 20]. Among them, recent research, such as BOTFINDER and DISCLOSURE, often used the NetFlow data, which provides an abstraction of TCP/UDP flows. However, we already confirmed that the regular access pattern is not only observed in C&C but also observed in many benign applications. Namely, studies that focused on the regular access pattern actually detected a simple programmatic behavior, including many benign automatic activities. To overcome this shortcoming, some researchers said that the white list or reputation system can cover the blind spot.

Specifically, BOTFINDER used a whitelist to pre-filter the bot-like benign traffic. And DISCLOSURE used a number of external reputation scores for its classifier. In both works, the goal of using whitelist is to reduce the false positives.

However, there are also risks to generate false negatives. In one effort to avoid reputation-based filtering, some attackers either purchased or rented blocks of IP space with good reputations that were built up over the course of several years. Thus, the attackers had acquired known good IP blocks and used them for C&C servers. In this case, the effectiveness of the reputation system might not be guaranteed. In this work, we did not consider any dependencies on whitelists. We provide new distinctive features to distinguish HTTP-based C&Cs and archive a high accuracy of more than 96% and a low false positive rates of 1.3% even without using whitelists.

Group Behavior Analysis Approach. To maximize the damage, the attacker tries to infect as many hosts as possible. For example, bots have a self-propagating code designed to infect hosts in the same network. Using this tendency, group behavior analysis approaches [4, 8] are appeared. However, this tendency may not be seen prominently in a small network or those detection systems cannot detect a single infected host.

Payload Analysis Approach. There have been several studies [2, 7, 13, 22] to identify the abnormal payload generated by malware. They focused on the difference in byte frequency between normal payloads and malicious payloads. In BotHunter [7], they measured the Mahalanobis distance of “Z-string,” which is the rank-ordered byte frequency of a payload similar to PAYL [22]. However, they mainly focused on detecting IRC-based C&Cs, TCP-based C&Cs or buffer overflow attacks rather than the HTTP-based C&Cs. TAMD [25] used both the group behavior analysis approach and the payload analysis approach. One of their premises was that the C&C often carries similar payloads. However, HTTP-based bots typically send different identification tokens or different payloads at each time. Thus, the payloads might not be similar even though a number of the same bots are spread in the same network. In contrast, our approach provides novel criteria, which are the low density and the high content variability, for identifying HTTP-based C&C.

8. Discussion

The Pros and Cons of Using HAS. Here we discuss the pros and cons of using a HAS compared with using NetFlow. A NetFlow is defined as an abstracted unidirectional sequence of packets that share specific network properties (e.g., IP source/destination addresses, and TCP or UDP source/destination ports), while a HAS is defined as an abstracted bidirectional

series of web requests generated by a heuristic methodology as introduced in **Fig. 3**. In order to extract a HAS from network traffic, Deep Packet Inspection is required. Thus, the extraction of a HAS consumes more time than that of NetFlow. Further, we cannot extract a HAS from tunneled traffic such as HTTPS using TLS/SSL, whereas NetFlow identifies even tunneled C&Cs. However, HAS has an advantage for analyzing the network activity of applications. The HTTP connections between the same end point A and B can have varying source ports, whereas the destination port stays the same. Therefore, for an application activity, NetFlow generates non-correlated multiple flows, whereas the HAS extraction generates only one HAS that is tightly correlated.

Remark. In our current work, we decided to determine τ through an empirical way. We measured the maximum interval of HTTP requests by visiting top 100 most-visited sites, among the experimental data in Trace-B, 100 times. When we selected Tau as 1.8 second, slightly longer than the maximum time interval, most of the web connections could be merged into HASes accurately. Also, few connections did not belong to any HASes because they didn't have referer information neither have the same 2nd-level domain name. Though we determined the parameter τ through the heuristic way, it still requires an advanced method. It might be a future work.

Limitation. One method an attacker might use is to utilize a large traffic volume to communicate with multiple C&C servers. However, this method degrades the stealth of the malware, and using multiple C&C servers requires increased costs for attackers. Another approach is to make content heterogeneity forcibly by putting the same strings or binaries in addition to a series of malicious content. This technique might degrade the overall accuracy. We developed our custom info-stealing malware that exfiltrates stolen data padded with constant text data. The total size of each connection had over 20 KB to evade the low density check. Even though the real data changed time to time, the HAS-Analyzer could not detect them. Though we did not find such malware yet, content heterogeneity technique might appear soon. To overcome this limitation, we can re-train the HAS-Analyzer with C&C activities using content heterogeneity. In addition, we should find more precise features of C&C activities.

Privacy Concern. Though we used hash-based sanitization for the content of web connections, the TCP header values still remained exposed. We can also import hash-based sanitization [15] for hiding sensitive information of TCP headers, such as the source IP address, in order to preserve privacy. The mapping table between hash values and the real source IP addresses need to be managed by operators.

9. Conclusion

We introduced new features of HTTP-based C&C activities based on the analysis of the HAS. The HAS is a useful data structure for analyzing network activities of HTTP-based applications. By analyzing multiple statistics of the HAS, we identified C&C activities with a high accuracy and a low false positive rate. The effectiveness of the feature sets was proved in the feature evaluation. We are going to apply our proposed system, the HAS-Analyzer, to a real network environment, that is, an ISP.

References

[1] Leyla Bilge, Davide Balzarotti and William Robertson, "DISCLOSURE: Detecting Botnet Command and Control Servers Through Large-Scale NetFlow Analysis," in *Proc. of 2012 Annual Computer Security Applications Conference (ACSAC)*, pp. 129-138, December 3-7, 2012. Article (CrossRef Link)

[2] James R. Binkley and Suresh Singh, "An Algorithm for Anomaly-based Botnet Detection," in *Proc. of the 2nd conf. on Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, vol. 2, pp. 7, July 7, 2006. Article (CrossRefLink)

[3] B. Claise, "Cisco Systems NetFlow Services Export Version 9. RFC 3954," *Internet Engineering Task Force*, 2004.

[4] Hyunsang Choi, Heejo Lee and Hyogon Kim, "BotGAD: detecting botnets by capturing group activities in network traffic," in *Proc. of 4th International ICST Con. on COMmunication System softWAre and middleware* (*COMSWARE*), Article No. 2, June 16-19, 2009. Article (CrossRef Link)

[5] Christian J. Dietrich, Christian Rossow and Norbert Pohlmann, "CoCoSpot: Clustering and recognizing botnet command and control channels using traffic analysis," *The Int. Journal of Computer and Telecommunications Networking*, vol.57, issue.2, pp. 475-486, Feb. 2013. Article (CrossRef Link)

[6] Guofei Gu, Roberto Perdisci, Junjie Zhang and Wenke Lee, "BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *Proc. of 17th conf. on Security symposium*, pp. 139-154, July 28 - August 1, 2008. Article (CrossRefLink)

[7] Guofei. Gu, Phillip. Porras, Vinod. Yegneswaran, Martin. Fong and Wenke. Lee, "Bothunter: Detecting Malware Infection through IDS-drivn dialog correlation," in *Proc. of 16th USENIX Security Symposium*, Article No. 12, August 6-10, 2007. Article (CrossRefLink)

[8] Guofei Gu, Phillip Porras, Vinod Yegneswaran, "BotSniffer: Detecting Botnet Command and Control Channels," in *Proc. of 15th Annual Network and Distributed System Security Symposium* (*NDSS*), 2008. Article (CrossRefLink)

[9] Jason Jones, "State of Web Exploit Kits," *Black Hat USA*, 2012. Article(CrossRefLink)

[10] Anestis Karasaridis, Brian Rexroad and David Hoein, "Wide-scale Botnet Detection and Characterization," in *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets (HotBots)*, pp. 7, April 10, 2007. Article (CrossRefLink)

[11] Andy Liaw and Matthew Wiener, "Classification and regression by random forest," *R News*, vol. 2, num.3, pp.18-22, Dec. 2002. Article (CrossRefLink)

[12] Carl Livadas , Robert Walsh , David Lapsley and W. Timothy Strayer, "Using machine learning techniques to identify botnet traffic," in *Proceedings 2006 31st IEEE Conference on Local Computer Networks (LCN)*, pp. 967-974, November 14-16, 2006. Article (CrossRef Link)

[13] Wei Lu, Mahbod Tavallaee and Ali 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)*, pp. 1-10, March 10-12, 2009. Article (CrossRef Link)

[14] Malware Domian List," in http://www.malwaredomainlist.com/

[15] Janak J. Parekh, Ke Wang and Salvatore J. Stolfo, "Privacy Preserving Payload based Correlation for Accurate Malicious Traffic Detection," in *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense (LSAD)*, pp. 99-106, September 11, 2006. Article (CrossRef Link)

[16] Roberto Perdisci, Wenke Lee and Nick Feamster, "Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces," in *Proceedings of the 7th USENIX conference on Networked systems design and implementation (NSDI)*, pp. 26, April 28-30, 2010. Article (CrossRefLink)

[17] J. Ross Quinlan, "C4.5: Programs for machine learning," *In Morgan Kaufmann Publishers*, 1993.

[18] Caitlin Sadowski and Greg Levin, "SimHash: Hash-based Similarity Detection," *http://simhash.googlecode.com/*, 2007.

[19] Elizabeth Stinson and John C. Mitchell, "Towards systematic evaluation of the evadability of bot/botnet detection methods," in *Proc. of Proceedings of the 2nd conference on USENIX Workshop on offensive technologies (WOOT)*, Article No. 5, July 28, 2008. Article (CrossRefLink)
[20] Florian Tegeler, Xiaoming Fu, Giovanni Vigna and Christopher Kruegel, "BotFinder: Finding Bots in Network Traffic Without Deep Packet Inspection," in *Proc. of The 8th International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, pp. 349-360, December 10-13, 2012. Article (CrossRef Link)
[21] Virustotal intelligence" in https://www.virustotal.com/intelligence/
[22] Ke Wang and Salvatore 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. Article (CrossRef Link)
[23] "Weka 3: Data Mining Software in Java," *in http://www.cs.waikato.ac.nz/ml/weka/*
[24] Peter Wurzinger, Leyla Bilge, Thorsten Holz, Jan Goebel and Christopher Kruegel, "Automatically Generating Models for Botnet Detection," in *Proc. of Proceedings of the 14th European conference on Research in computer security (ESORICS)*, pp. 232-249, September 21-25, 2009. Article (CrossRef Link)
[25] Ting-Fang Yen and Michael K. Reiter, "Traffic Aggregation for Malware Detection," *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, pp. 207-227, July 10-11, 2008. Article (CrossRef Link)
[26] H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. Youssef, M. Debbabi, and L. Wang, "On the Analysis of the Zeus Botnet Crimeware Toolkit," in *Proc. of International Conference on Privacy, Security and Trust*, pp. 31-38, August 17-19, 2010. Article (CrossRef Link)

Sung-Jin Kim is received the M.S. degree in Computer and Communications Engineering from the Pohang Univaersity of Science and Technology, Korea. He is now a member of engineering staff at the Attached Institute of ETRI in Korea. His research interests include network security, with a focus on malicious traffic analysis, botnet and APT research.

Sungryoul Lee received his B.S. and M.S. from Dept. of Computer Science, Sogang Universtiy, Korean, in 2001 and 2003, respectively, and his Ph.D. from the School of Electrical Engineering and Computer Sciences, Seoul National University, Korea, in 2010. Now, he work for the Attached Institute of ETRI in Korea. His research interests include Power Saving Technologies for Wireless Networks, Network Security, Information Security.

Byungchul Bae received his B.S. and M.S. in Computer Science from the Hongik Univaersity, Korean. He is now a principle member of engineering staff at the Attached Institute of ETRI in Korea. He has been working on distributed computing and network security, with a focus on cyber security monitoring and control.