Master's Thesis

Title

BotProfiler++: Detecting Malware-Infected Hosts using Templates of Time-Series HTTP Request Patterns

Supervisor

Professor Masayuki Murata

Author

Taiga Hokaguchi

February 8th, 2019

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

BotProfiler++: Detecting Malware-Infected Hosts using Templates of Time-Series HTTP Request Patterns

Taiga Hokaguchi

Abstract

Malware-infected hosts are one of the most serious threats in network services. An attacker controls many malware-infected hosts via command and control (C&C) servers to carry out cyber attacks. The victim of the attack observes only the packets from the malware-infected hosts. Thus, it is difficult to identify the attacker.

One approach to mitigating such attacks is to detect malware-infected hosts. BotProfiler is one of the methods that detect malware-infected hosts. BotProfiler focuses on the communications between malware-infected hosts and C&C servers. Most malware use HTTP as their C&C protocol. Thus, BotProfiler generates the templates of HTTP requests sent by malware-infected hosts and detects malware-infected hosts by comparing the monitored HTTP requests and the templates. BotProfiler avoids misdetections by introducing the rarity which is calculated by the frequency of the HTTP requests monitored at the deployment network. However, BotProfiler may cause a large number of misdetections before a sufficient amount of benign traffic are monitored. Moreover, even if a sufficient amount of benign traffic to calculate rarity are monitored, the rarity may cause the undetected C&C communications; the C&C communications whose rarity scores are low are not detected even if they exactly match the templates of the C&C communications.

In this thesis, we propose a method that detects malware-infected hosts with high detection rate and low false detection rate without using the data on benign communications. Based on the fact that many malware-infected hosts generate multiple HTTP requests, we propose a method using the templates of the sets of the HTTP requests. This method generates a template called group template for each malware that is constructed of the set of templates of HTTP requests the malware generates. Then, it detects malware-infected

hosts by comparing the set of monitored HTTP requests with the group templates. Because the probability that the set of benign traffic matches the group templates is low, our method rarely misdetects the benign traffic even though our method does not use the data on the monitored benign traffic.

We implement our method and evaluate it using real traffic data. The results show that our method detects 93.22% of malware-infected hosts with only 3% of false positive ratio even when we do not use the data on benign traffic monitored in advance.

Keywords

Cyber Security

Malware

Detection

Bot

Template

Contents

List of Figures

List of Tables

1 Introduction

Malware-infected hosts are one of the most serious threats in network services. An attacker controls many malware-infected hosts via command and control (C&C) servers to carry out cyber attacks [1,2]. An infected host controlled by C&C is called a bot, and a group of bots connected via C&C is called a botnet. An Attacker sends attack instructions to bots to carry out attacks and receives the results of attacks by bots via C&C. The victim of the attack observes only the packets from the malware-infected hosts. Thus, it is difficult to identify the attacker.

One approach to stopping such attacks is to defeat the botnets [3]. Defeating the botnets prevents cyber attacks. However, it is difficult to defeat the botnets, because the precise track of the C&C servers and the cooperation among the relevant organizations are necessary.

Another approach is to detect malware-infected hosts. There are two types of methods to detect malware-infected hosts: host-based and network-based methods. A host-based method monitors the processes and detects malware by installing programs such as antivirus software. However, such programs may be disabled by the attackers if the host is controlled by attackers. Thus, network-based approach is effective to detect malware infected hosts, and many network-based methods have been proposed [4–6].

Kuhrer et al. proposed a blacklist system that detects C&C communication by blacklisting the known domains including C&C servers [7]. This method detects the packets to the blacklisted domains. However, attackers frequently change the domains of the C&C servers to avoid detection by the blacklist [8]. Thus, the blacklist is not always successful.

Methods using the template of the C&C communication instead of the blacklists have also been proposed [9, 10]. Many botnets use HTTP as their C&C protocol. Thus, these methods generate the template of HTTP requests that malware-infected hosts generate. These methods monitor the HTTP requests sent from the monitored network and detect the C&C communication based on the similarity from the template. There may be the benign traffic similar to the template of the C&C communications. Considering such benign traffic, they introduce a metric called rarity that indicates the infrequency of the HTTP requests in the monitored network. That is, they calculate the similarity score

considering both of the rarity of the current request and the similarity between the current request and template, and detect C&C communications if the score exceeds a predefined threshold. Another method to avoiding misdetection of the benign traffic similar to the template is to generate the template considering the legitimate traffic. Mizuno et al. proposed a method to generate the template by machine learning using both of the C&C communications and the monitored benign traffic as training data [11].

The methods to avoid the misdetection of benign traffic requires a sufficient amount of monitored benign traffic. They may cause a large number of misdetections before a sufficient amount of benign traffic are monitored. Moreover, even if a sufficient amount of benign traffic to calculate rarity are monitored, the rarity may cause the undetected C&C communications; the C&C communications whose rarity scores are low are not detected even if they exactly match the templates of the C&C communications.

In this thesis, we propose a method that detects malware-infected hosts with high detection rate and low false detection rate without using the data on benign communications. Based on the fact that many malware-infected hosts generate multiple HTTP requests, we propose a method using the templates of the sets of the HTTP requests. This method generates a template called group template for each malware that is constructed of the set of templates of HTTP requests the malware generates. Then, it detects malware-infected hosts by comparing the set of monitored HTTP requests with the group templates. Because the probability that the set of benign traffic matches the group templates is low, our method rarely misdetects the benign traffic even though our method does not use the data on the monitored benign traffic. In this thesis, we implement our method and evaluate its detection performance.

The rest of this thesis is organized as follows. Section 2 introduces an existing method to detect malware-infected hosts based on the templates of HTTP requests. In Section 3, we analyze the HTTP requests generated by the malware-infected hosts. We propose a method to detect malware-infected hosts based on the group template in Section 4. We evaluate our method in Section 5. Section 6 discusses the advantages and limitations of our method. Section 7 reviews related work. Finally, Section 8 concludes the thesis.

2 BotProfiler: Existing Method to Detect Malware-Infected Hosts using Templates of an HTTP Request

Chiba et al. proposed a system called BotProfiler [10]. BotProfiler generates templates to detect infected hosts in a network. In this section, we first describe overview of BotProfiler, and then explain the template generation method and template matching method.

2.1 System Overview

BotProfiler generates templates based on HTTP requests sent by the previously monitored malware-infected hosts, and detects infected hosts by comparing the newly observed HTTP requests with the templates. Figure 1 shows an overview of BotProfiler. It involves four steps; step 1: variability profiling, step 2: template generation, step 3: rarity profiling, and step 4: template matching. Steps 1 and 2 generate templates from outbound traffic captured in sandbox system running malware samples. Steps 3 and 4 are performed at the deployment network to detect malware-infected hosts.

2.2 Template Generation

2.2.1 Step 1: Variability Profiling

BotProfiler generates templates of an HTTP request sent by the malware-infected hosts. Malicious infrastructures, such as malware and C&C, tend to be reused instead of created from scratch. As a result, some substrings in HTTP requests are common to multiple malwares. Therefore, BotProfiler extracts such a common substrings as invariable keywords. First, substrings composed of two or more characters in URL paths, URL queries, and user agents in HTTP requests captured in sandbox system running malware-infected hosts are extracted as candidate keywords. From candidate keywords, invariable keywords are then identified based on the number of malware samples using the keywords.

2.2.2 Step2: Template Generation

Step 2 generates templates using invariable keywords produced in Step 1. First, HTTP requests are segmented into substrings with symbols (e.g., ‘/’, ‘?’, ‘=’, ‘-’, ‘.’) and

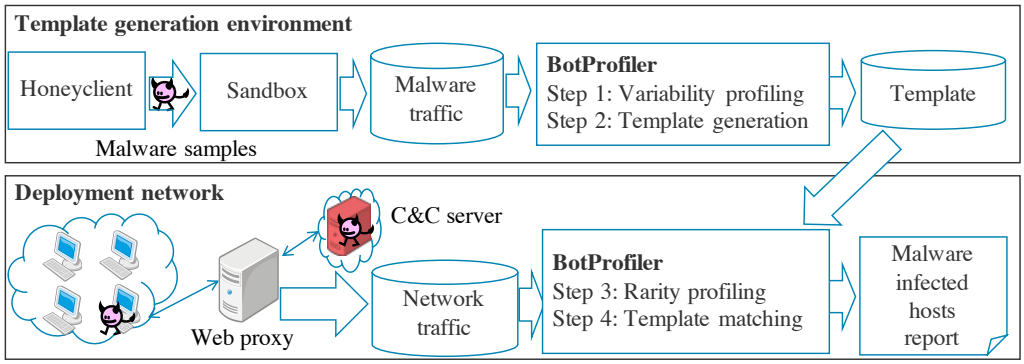

Figure 1: Overview of BotProfiler.

Table 1: Example of patterns in regular expressions.

Data type	Regular expression
String	<str; *length*>
Integer	<int; *length*>
Hexadecimal	<hex; *length*>
Base64	<base64; *length*>

replaced with regular expressions (e.g., <str; *length*>) containing the data type (e.g., string (str), integer (int), hexadecimal(hex), base64) and length indicated in Table 1.

After the replacement, similar HTTP requests are aggregated into one template. Specifically, it runs two stage of clustering of HTTP requests, and extracts represent one in each cluster as template. The first stage makes clusters based on the destination IP addresses. This clustering process makes IP range clusters by grouping HTTP requests whose destination IP addresses or destination IP prefixes are the same. The second stage makes clusters based on the similarity between HTTP requests within each IP range cluster. In this clustering, we use the similarity metric $Sim\,(h_a, h_b)$ between HTTP requests h_a and h_b that is defined by the following equation.

$$Sim\,(h_a, h_b) = \frac{1}{n} \cdot \sum_{k=1}^{n} \sigma_k\,(h_a, h_b) \tag{1}$$

Here, σ_k is the function to calculate similarities between elements in h_a and h_b, and n is the number of considered elements; we set $n = 4$. Specifically, σ_1 is the similarity

between URL paths and is calculated using the normalized edit distance, σ_2 is the similarity between the combination of parameter names in the URL queries and is calculated using the Jaccard similarity, σ_3 is the similarity between the values in the URL queries and is calculated using the ratio of having the same data type and length, and σ_4 is the similarity between user agents and is calculated using the normalized edit distance. The above definitions result in $\sigma_k \in [0,1], \forall k$ and $Sim(h_a, h_b) \in [0,1]$.

After two stage of clustering, BotProfiler extracts the centroid in each clusters. Specifically, it extract the centroid which refers to one of the HTTP requests that maximizes the sum of similarities between the request and all other requests. From the centroids, the templates that contain URL paths, URL queries, and useragents are extracted.

2.3 Template Matching

2.3.1 Step3: Rarity Profiling

Step 3 calculate the rarity that indicates the infrequency of the HTTP request monitored at the deployment network. BotProfiler focused on the rarity about URL paths, URL queries, and user agents. The rarity $\rho_{t,k}$ of an element k in a template t is calculated using the following equation.

$$\rho_{t,k} = 1 - \frac{n_{t,k}}{max_i n_i} \tag{2}$$

Here, $n_{t,k}$ is the number of hosts that send HTTP requests containing k in t, and $max_i n_i$ is the maximum number of hosts in all elements of the same type (e.g., URL paths, URL queries, and user agents). The definition results in $\rho_{t,k} \in [0,1], \forall t, \forall k$.

2.3.2 Step4: Template Matching

Step 4 detects malware-infected hosts based on the matching score. Specifically, a matching score $Score(h,t)$ between an HTTP requet h and a template t is calculated from the following equation.

$$Score(h,t) = \frac{\sum_{k=1}^{n} \sigma_k(h,t) \cdot \omega(\sigma_k(h,t), \rho_{t,k})}{\sum_{k=1}^{n} \omega(\sigma_k(h,t), \rho_{t,k})} \cdot \rho_{h,d} \tag{3}$$

Here, $\sigma_k(h,t)$ is defined in the same way as explained in Section 2.2.2, $\rho_{t,k}$ is the rarity of k in t. $\rho_{h,d}$ is the rarity of a destination fully qualified domain name (FQDN) d in h and is calculated in the same way as step 3, and ω is the weight function between $\sigma_k(h,t)$ and $\rho_{t,k}$ and is defined by the following equation.

$$\omega\left(\sigma_k(h,t), \rho_{t,k}\right) = 1 + \frac{1}{\left(2 - \sigma_k(h,t) \cdot \rho_{t,k}\right)^m} \tag{4}$$

Here, m is a fixed parameter and determined empirically. The above definitions result in $Score(h,t) \in [0,1]$. $Score(h,t)$ is designed to be high when the similarity between an HTTP request h and a template t is high, and the rarities of elements in t are high in a deployment network. If $Score(h,t)$ exceeds a predefined threshold, which means an HTTP request closely matches a template and the elements in the request have rarely appeared in the deployment network, BotProfiler determines h to be generated by an infected host.

3 Analysis of Malicious Traffic

In this section, we investigate the traffic sent by the malware-infected hosts. Malware traffic was captured from the sandbox system running malware samples. The malware samples were collected using the honyeclient crawling public blacklists such as MDL [12] and hpHosts [13] and some commercial blacklists.

Figure 2 shows the distribution of the number of HTTP requests sent by each malware-infected host. The x-axis is the number of HTTP requests sent by each malware-infected host, and y-axis is the number of hosts. Figure 2 omits the points of more than 20 HTTP requests, because only 3% of the malware-infected hosts generate more than 20 HTTP requests. Figure 2 shows that more than 90% of malware-infected hosts send multiple HTTP requests.

Figure 3 shows the distribution of the time between the first and the last packets sent by each malware-infected host. The x-axis is the time between the first and the last packets sent by each malware-infected host, and the y-axis is the number of malware-infected hosts. Figure 3 omits the points of more than 30 seconds, because only less than 10% of the malware-infected hosts generates HTTP requests for more than 30 seconds. Figure 3 shows that most of the malware-infected hosts sent multiple HTTP requests within 30 seconds. That is, we can obtain the features of multiple HTTP requests sent by malware-infected hosts by monitoring HTTP requests for 30 seconds.

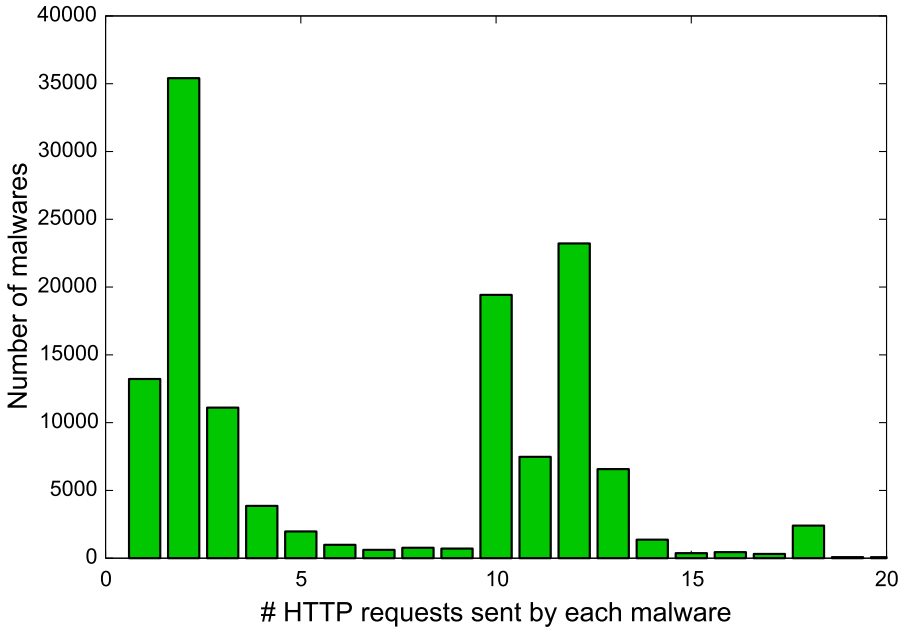

Figure 2: Distribution of the number of HTTP requests sent by each malware.

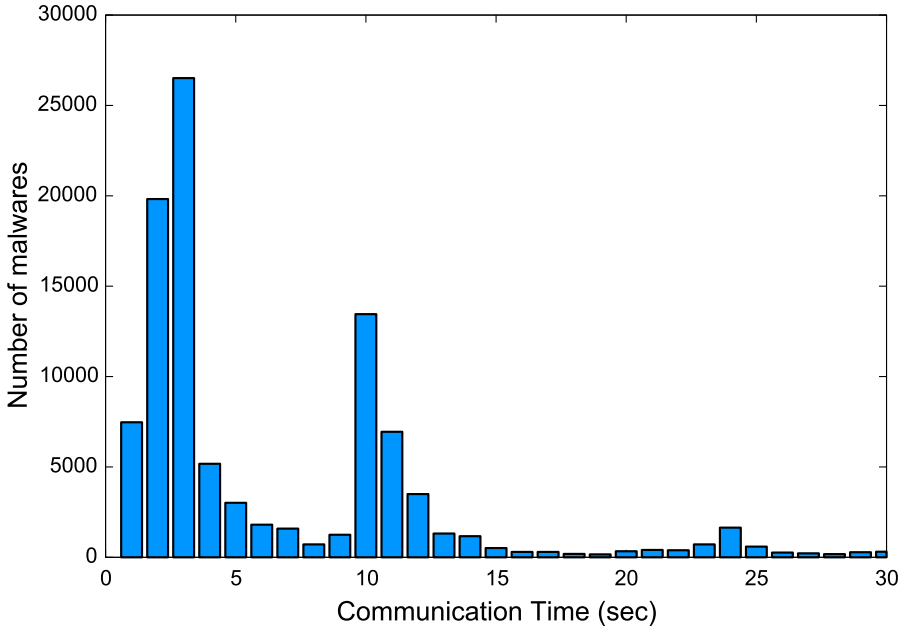

Figure 3: Distribution of the time between first and last packets sent by each malware.

4 BotProfiler++: New Method to Detect Malware-Infected Hosts using Templates of Time-Series Access Patterns

BotProfiler detects malware-infected hosts using the similarity between an HTTP request and a template and the rarities of elements. However, Botprofiler may misdetect a large amount of benign traffic before a sufficient amount of benign traffic are monitored. Moreover, even if a sufficient amount of benign traffic to calculate rarity are monitored, the rarity may cause the undetected C&C communications; the C&C communications whose rarity scores are low are not detected even if they exactly match the templates of the C&C communications.

Therefore, we propose a method that detects malware-infected hosts with high detection rate and low false detection rate without using the data on benign communications. Based on the fact that most of malware-infected hosts generate multiple HTTP requests, our method is based on the template for the set of multiple HTTP requests. This method generates a template called group template for each malware that is constructed of the set of templates of HTTP requests the malware generates. Then, it detects malware-infected hosts by comparing the set of monitored HTTP requests with the group templates. Because the probability that the set of benign traffic matches the group templates is low, our method rarely misdetects the benign traffic even though our method does not use the data on the monitored benign traffic.

4.1 System Overview

Figure 4 is an overview of our system. This system generates and uses the group template in addition to the single template generated by the BotProfiler. In our system, a group template T is defined by the set of single templates U_T that matches the of the HTTP requests sent by the malware-infected host, and the number of matched HTTP requests for each single template t_i ($i \in U_T$).

The details of our method is explained in the rest of this section.

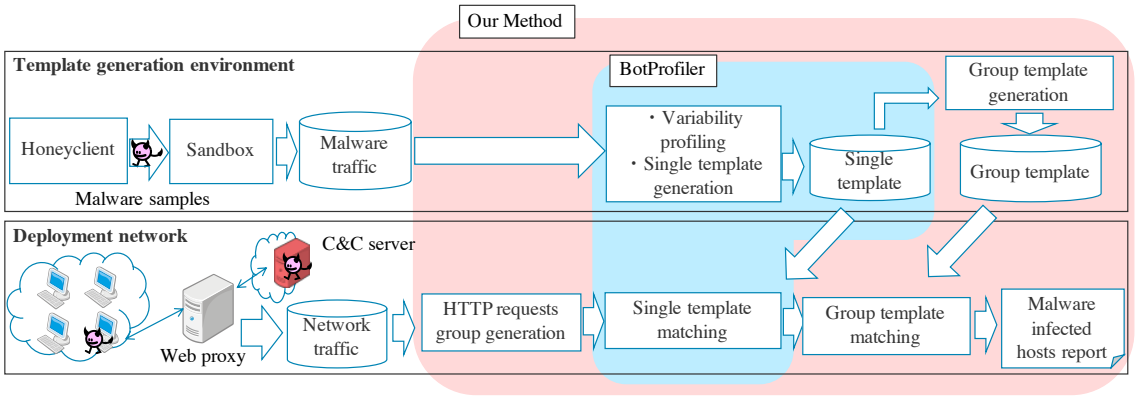

Figure 4: Overview of our system.

4.2 Group Template Generation

Our method first generates the single template by the same way as the Botprofiler using the malware traffic captured at the sandbox running malware-infected hosts. Then, based on the generated single template, we generate the group template by using the same malware traffic.

A group template are generated by the following steps for each malware-infected hosts. First, we select the single templates for each HTTP request sent by the host whose similarity is the highest. The selected templates are added to U_T. Then, we count the number of matched HTTP request t_i for $i \in U_T$.

4.3 Detection

4.3.1 HTTP Requests Group Generation

We first divide the time series of the HTTP requests into HTTP request groups, which are compared with the group templates. In this thesis, we divide the HTTP requests based on time. That is, the HTTP requests sent within a predefined time length are grouped into a HTTP request group. By setting the time length to a sufficiently large value, each HTTP request group includes a sufficient number of HTTP requests to capture the features of the HTTP requests sent by the malware-infected hosts.

4.3.2 Single Template Matching

Before using the group template, we first use the single template generated by BotProfiler to detect malware-infected hosts. However, unlike BotProfiler, we do not use the rarity, because we assume that a sufficient amount of benign traffic to calculate the rarity is not monitored. Therefore, we use the matching score $Score(h,t)$ defined by

$$Score(h,t) = \sum_{k=1}^{n} \sigma_k \tag{5}$$

If $Score(h,t) \leq \theta_L$ for all HTTP requests in the HTTP requests group, the HTTP requests group is decided as benign, because all of the HTTP requests does not match any single template. If $Score(h,t) \geq \theta_H$ for any HTTP requests in the HTTP requests group, the HTTP requests group is decided as malicious, because there are HTTP requests that exactly match the features of the HTTP requests generated by the malware-infected hosts. Otherwise, we perform the group template matching.

4.3.3 Group Template Matching

We detect the malware-infected hosts by comparing the HTTP request groups with the group templates. To compare them, we define the matching score $S(D,T)$ between an HTTP request group D and a group template T, and detect malware-infected hosts when $S(D,T)$ for one of group templates exceed the predefined threshold θ_G.

Though more sophisticated definition of $S(D,T)$ may exist, we simply define $S(D,T)$ by

$$S(D,T) = 1 - \frac{1}{|U_T|} \sum_{i \in U_T} s(d_i, t_i) \tag{6}$$

$$s(d_i,t_i) = \begin{cases} \alpha & (d_i = 0) \\ \frac{\beta(t_i - d_i)}{t_i} & (0 < d_i \leq t_i) \\ 0 & (d_i > t_i) \end{cases} \tag{7}$$

Here, D is a HTTP requests group, and T is a group template. d_i is the number of HTTP requests in D that whose score $Score(h,t)$ for single template i exceeds threshold

θ_L. α and β are fixed parameters and $\alpha >> \beta$. $S(D, T)$ becomes large as more single template in the group template T matches the HTTP request in the HTTP request group D.

By comparing HTTP request groups with the group template, our method accurately detects malware-infected hosts even when a sufficient amount of benign traffic to calculate the rarity is monitored, because the probability that multiple benign HTTP requests match the single template included in the group templates is small.

5 Evaluation

5.1 Data

Malware traffic was captured from the sandbox system [14] running malware samples. The sandbox supports executable files only in Microsoft Windows environments. These malware samples were collected using the honyeclient crawling public blacklists such as MDL [12] and hpHosts [13] and some commercial blacklists. Benign traffic was captured in an university. We divide malware samples and benign traffic into training data and testing data according to the date when the sample was collected. Table 2 shows the numbers of malicious HTTP requests and benign HTTP requests. Note that the benign HTTP requests in the training data set is not used by our method and is used only by BotProfiler.

In our method, HTTP requests are divided into HTTP request groups. In this thesis, we group the HTTP requests sent within 30 seconds from the first request into an HTTP request group, because the discussion on Section 3 indicates that most of malware-infected hosts generate multiple HTTP requests within 30 seconds.

Table 3 shows the benign HTTP request groups generated from the test data shown in Table 2. In this evaluation, we define the False Positive Rate (FPR) by the ratio of the number of the detected benign HTTP request groups to the total number of generated HTTP request groups shown in Table 3. Table 3 also shows the number of malware-infected hosts in the test data. In this evaluation, we define the True Positive Rate (TPR) by the ratio of the number of the detected malware-infected hosts to the total number of tested malware-infected hosts.

5.2 Results

Figure 5 shows the relationship between TPR and FPR. In this figure, we plot the TPR and FPR of our method by changing θ_G from 0.0 to 1.0 by 0.01. This figure also includes the results for the method that performs only the simple template matching using Eq. (5). We call this method BotProfiler without RP: Rarity Profiling. We also plot the TPR and FPR by changing θ_H from 0.0 to 1.0 by 0.01.

Figure 5 indicates that our method achieves higher TPR and lower FPR than Bot-

Table 2: Dataset.

	Training		Testing	
Label	Period	# HTTP requests	Period	# HTTP requests
Malicious	2017/8/1 - 2017/12/31	656,714	2018/1/1 - 2018/3/31	442,532
Benign	2018/12/1 - 2018/12/31	291,343	2018/1/1 - 2018/3/31	876,778

Table 3: The number of malware-infected hosts and the number of benign HTTP request groups in test data.

Malware-infected hosts	Benign HTTP requests groups
44,116	30,863

Profiler without RP. This is because our method uses the group template. The group template matching detects malware-infected hosts even if HTTP requests match multiple single templates, which avoids misdetections.

To discuss the details of results, we investigates the TPR and FPR of our method when θ_G is changed. Table 4 shows the FPR and TPR when $\theta_H = 0.90$ and $\theta_L = 0.80$. Table 4 indicates that θ_G has only a small impact on the TPR and the FPR. This is because the matching score for group template does not become larger than 0.75 unless multiple single templates are matched. That is, by setting θ_G to the value more than 0.75, only HTTP request groups including multiple HTTP requests matches different single templates are detected. As a result, we avoid misdetection.

Table 5 investigates the impact of θ_H and θ_L on the TPR and the FPR. In Table 5, we set $\theta_G = 0.85$. Table 5 indicates that a small θ_H causes many misdetection. That is, θ_H should be set to a large value. Table 5 indicates that a small θ_L makes TPR high even if θ_H is set to a large value. Therefore, our method achieves a high TPR and a small FPR by setting θ_H to a large value and θ_L to a small value.

We also compare our method with BotProfiler using rarity profiling. Figure 6 compares the TPR and FPR, similar to Figure 5. Figure 6 indicates that BotProfiler achieves very low FPR. That is, the rarity profiling is useful to avoid misdetection. However, BotProfiler cannot achieve TPR more than 87.17%. This is because the malicious HTTP requests similar to the benign HTTP requests are not detected by BotProfiler. On the other

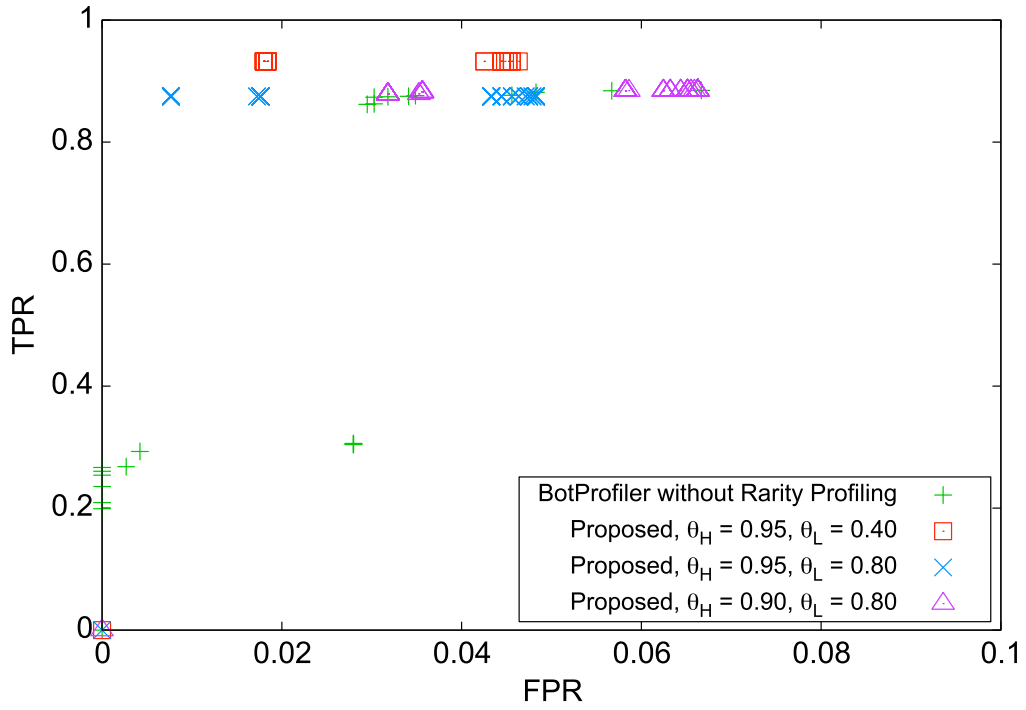

Figure 5: Relationship between TPR and FPR of BotProfiler without RP and our method.

Table 4: TPR and FPR of our method when θ_G is changed with $\theta_H = 0.90$ and $\theta_L = 0.80$.

θ_G		0.90	0.85	0.80	0.75
Proposed	TPR	87.87%	87.87%	87.88%	87.91%
	FPR	3.18%	3.18%	3.52%	3.56%

hand, our method achieves TPR more than 87.17%. This is because our method detects malware-infected hosts when multiple single templates matches. As a result, our method detects infected hosts even when HTTP requests from the infected hosts are similar to the benign HTTP requests.

To simply compare our method with BotProfiler and BotProfiler without RP, we compare the TPRs when the thresholds are set so as to make the FPR less than 3% in Table 6. Our method with $\theta_L = 0.80$ and $\theta_H = 0.90$ cannot achieve FPR less than 3%. Table 6 indicates that our method with $\theta_L = 0.40$ and $\theta_H = 0.95$ achieves the highest TPR.

Table 5: TPR and FPR when changing θ_H and θ_L by the our method.

θ_H	θ_L	TPR	FPR
0.95	0.40	93.22%	1.80%
0.95	0.80	87.49%	0.76%
0.90	0.80	87.87%	3.18%

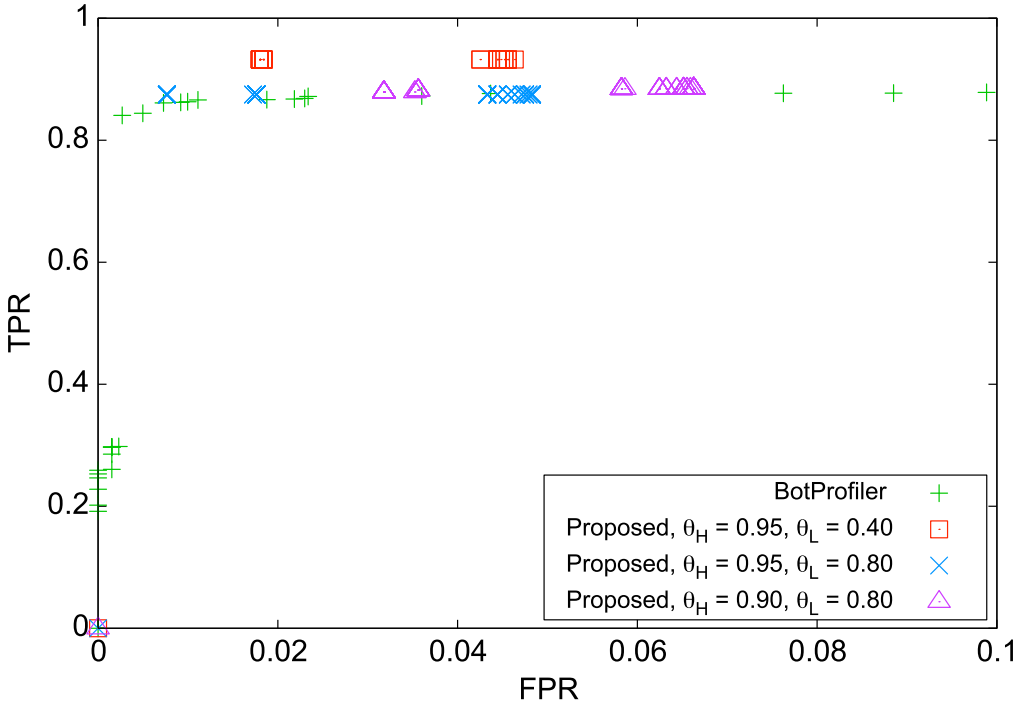

Figure 6: Relationship between TPR and FPR of BotProfiler and our method.

Table 6: TPR when parameters are set so as to make FPR less than 3%.

		TPR
BotProfiler without RP		86.18%
BotProfiler		87.17%
Our method	$\theta_H = 0.95, \theta_L = 0.40$	93.22%
	$\theta_H = 0.95, \theta_L = 0.80$	87.49%
	$\theta_H = 0.90, \theta_L = 0.80$	-

6 Discussion

The Case where Malicious and Benign Traffic are Mixed In our evaluation, we use the malicious traffic captured at the sandbox as the malicious test data. That is, the malicious test data does not include benign traffic. On the other hand, in the real environment, malware-infected hosts are also operated by the user and generate benign traffic in addition to the malicious traffic. Our method detects malware-infected hosts even in such cases because our method detects the infected hosts if there exists multiple HTTP requests matches the templates in the group template. That is, the benign traffic does not prevent the detection by our method.

HTTP Request Group Generation In our evaluation, we generate HTTP request groups by simply dividing the time series of the HTTP requests. But, HTTP request groups should be generated so as to include a sufficient number of HTTP requests to identify the requests sent by the infected hosts. One approach to generating HTTP request groups is a sliding window approach. In this approach, when a new HTTP request arrives, an HTTP request group that includes the HTTP request within s seconds are generated. And, if one of the generated HTTP request groups is detected as malicious, the malware-infected hosts are detected.

Parameter Tuning According to the results, the TPR and the FPR depend on the parameters. A suitable parameter may be set based on the malicious traffic collected at the sandbox; the parameter should be set to detect malicious traffic but the range that are regarded to malicious should be small so as to avoid misdetection. A method to set the suitable parameters is one of our future work.

7 Related Work

Various methods for malware-infected hosts have been proposed. Gu et al. proposed a detection method by modeling the relationship of common actions when hosts are infected by malwares [4]. However, this method is host-based, and the operation of the malware detection program may be disabled when malware-infected hosts are controled by attackers.

Therefore, many researchers focus on the network-based methods. Perdisci et al. proposed a method of generating signatures with regular expressions to detect the URLs used in C&C communications based on HTTP traffic captured in a controlled environment [5]. Nelems et al. proposed a system called ExecScent, which generates templates with regular expressions and detects malware-infected hosts using not only the similarity of templates but also rarity which indicates frequency of HTTP requests generally occurring in monitoring target network [9]. Chiba et al. improved Nelems et al.'s method, which focused on common substrings appears among multiple malware because malicious infrastructures, such as malware and C&C, tend to be reused instead of created from scratch [10]. This method is called BotProfiler. BotProfiler extracts these substrings as invariable keywords and generates templates with invariable keywords and regular expressions. Mizuno et al. proposed a method using machine learning. This method classifies traffic as malicious and benign. After classifying, it aggregates similar features and generates template.

These methods requires a sufficient amount of monitored benign traffic to avoid misdetection of benign traffic. They may cause a large number of misdetections before a sufficient amount of benign traffic are monitored. On the other hand, our method focuses on the fact that many malware-infected hosts generate multiple HTTP requests, and detects malware-infected hosts with high detection rate and low false detection rate without using the data on benign communications.

There are a method to detect malicious traffic by using multiple HTTP requests. We have proposed another method to detect drive-by download attacks [15], instead of focusing on the C&C communications. Drive-by download attacks are carried out by an attacker guiding from a tampering site through a multilevel redirect to the malware download site. This method learns such multilevel redirect structure as a feature of drive-by download

attacks. This method uses a series of HTTP requests included in drive-by download attacks and a series of benign HTTP requests that occur on a daily for learning. On the other hand, in this thesis, we focus on the C&C communications and generates group templates of multiple HTTP requests sent by malware-infected hosts. We demonstrate that our method using the group templates accurately detects malware-infected host even when the information on the benign traffic are not observed.

8 Conclusion

We proposed a method that detects malware-infected hosts with high detection rate and low false detection rate without using the data on benign communications. Based on the fact that many malware-infected hosts generate multiple HTTP requests, our method uses the templates of the sets of the HTTP requests. This method generates a template called group template for each malware that is constructed of the set of templates of HTTP requests the malware generates. Then, it detects malware-infected hosts by comparing the set of monitored HTTP requests with the group templates. Because the probability that the set of benign traffic matches the group templates is low, our method rarely misdetects the benign traffic even though our method does not use the data on the monitored benign traffic.

We implemented our method and evaluated it using real traffic data. The results show that our method detects 93.22% of malware-infected hosts with only 3% of false positive ratio even when we do not use the data on benign traffic monitored in advance. However, the true positive rate and false positive rate depend on the parameters. The parameter should be set to detect malicious traffic but the range that are regarded to malicious should be small so as to avoid misdetection. A method to set the suitable parameters is one of our future work.

Acknowledgments

This thesis would not accomplish without a lot of great support from many people. First, I would like to express my deepest gratitude to Professor Masayuki Murata of Osaka University, for providing me with the opportunity to research with a talented team of researchers. His creative suggestions, insightful comments, and patient encouragement have been essential for my research activity.

Furthermore, I would like to show my greatest appreciation to Associate Professor Yuichi Ohsita of Osaka University. He devoted a great deal of time for me and gave me a lot of advice about my research. Without his continuous support, my work would not be accomplished. In addition, he enthusiastically taught me the way of thinking and writing. They must be invaluable skills in the future of my life.

Moreover, I would like to show my appreciation to Associate Professor Shin'ichi Arakawa, Assistant Professor Daichi Kominami, Assistant Professor Naomi Kuze, and Specially Appointed Assistant Professor Tatsuya Otoshi of Osaka University for appreciated comments and support. Their kindnesses on my behalf were invaluable, and I am forever in debt.

I am grateful to Dr. Mitsuaki Akiyama, Dr. Daiki Chiba, and Mr. Toshiki Shibahara of NTT Corporation for providing me many valuable comments. The opportunity of discussion at the NTT Secure Platform Laboratories improved my explanation capability.

I am also grateful to Specially Appointed Professor Makoto Imase of Osaka University for his advice of my research and experimental environment. His advice from different perspective points of view was always informative and helpful.

I would like to thank all the members of Advanced Network Architecture Research Laboratory at the Graduate School of Information Science and Technology, Osaka University, for their inciting discussions and fellowship.

Last, but not least, I thank my parents and my sister for their invaluable support and constant encouragement during my master studies.

References

[1] M. Feily, A. Shahrestani, and S. Ramadass, "A Survey of Botnet and Botnet Detection," in *Proceedings of Third International Conference on Emerging Security Information, Systems and Technologies*. IEEE, 2009, pp. 268–273.

[2] A. Emigh, "The Crimeware Landscape: Malware, Phishing, Identity Theft and Beyond," *Journal of Digital Forensic Practice*, vol. 1, no. 3, pp. 245–260, 2006.

[3] R. Boscovich, "Microsoft and Financial Services Industry Leaders Target Cybercriminal Operations from Zeus Botnets," *The official Microsoft blog*, 2012.

[4] G. Gu, P. A. Porras, V. Yegneswaran, M. W. Fong, and W. Lee, "Bothunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation." in *Proceedings of USENIX Security Symposium*, vol. 7, 2007, pp. 1–16.

[5] R. Perdisci, W. Lee, and N. Feamster, "Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces." in *Proceedings of 15th USENIX Symposium on Networked Systems Design and Implementation*, vol. 10, 2010, p. 14.

[6] R. Bortolameotti, T. van Ede, M. Caselli, M. H. Everts, P. Hartel, R. Hofstede, W. Jonker, and A. Peter, "DECANTeR: DEteCtion of Anomalous outbouNd HTTP TRaffic by Passive Application Fingerprinting," in *Proceedings of the 33rd Annual Computer Security Applications Conference*. ACM, 2017, pp. 373–386.

[7] M. Kührer, C. Rossow, and T. Holz, "Paint it black: Evaluating the effectiveness of malware blacklists," in *Proceedings of International Workshop on Recent Advances in Intrusion Detection*. Springer, 2014, pp. 1–21.

[8] P. Vinod, R. Jaipur, V. Laxmi, and M. Gaur, "Survey on malware detection methods," in *Proceedings of the 3rd Hackers' Workshop on Computer and Internet Security*, 2009, pp. 74–79.

[9] T. Nelms, R. Perdisci, and M. Ahamad, "ExecScent: Mining for New C&C Domains in Live Networks with Adaptive Control Protocol Templates." in *Proceedings of USENIX Security Symposium*, 2013, pp. 589–604.

[10] D. Chiba, T. Yagi, M. Akiyama, K. Aoki, T. Hariu, and S. Goto, "BotProfiler: Detecting Malware-Infected Hosts by Profiling Variability of Malicious Infrastructure," *IEICE Transactions on Communications*, vol. 99, no. 5, pp. 1012–1023, 2016.

[11] S. Mizuno, M. Hatada, T. Mori, and S. Goto, "Botdetector: A robust and scalable approach toward detecting malware-infected devices," in *Proceedings of IEEE International Conference on Communications*. IEEE, 2017, pp. 1–7.

[12] "Malware Domain List." http://www.malwaredomainlist.com/, accessed December 10. 2018.

[13] "Malwarebytes," http://www.hosts-file.net/, accessed December 10. 2018.

[14] K. Aoki, T. Yagi, M. Iwamura, and M. Itoh, "Controlling Malware HTTP Communications in Dynamic Analysis System using Search Engine," in *Proceedings of Third International Workshop on Cyberspace Safety and Security*. IEEE, 2011, pp. 1–6.

[15] T. Shibahara, K. Yamanishi, Y. Takata, D. Chiba, T. Hokaguchi, M. Akiyama, T. Yagi, Y. Ohsita, and M. Murata, "Event De-Noising Convolutional Neural Network for Detecting Malicious URL Sequences from Proxy Logs," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 101, no. 12, pp. 2149–2161, 2018.