Hindawi
Security and Communication Networks
Volume 2021, Article ID 6653386, 12 pages
https://doi.org/10.1155/2021/6653386

Research Article

HTTP-Based APT Malware Infection Detection Using URL Correlation Analysis

Wei-Na Niu,[1] **Jiao Xie**,[1] **Xiao-Song Zhang**,[1,2] **Chong Wang**,[1] **Xin-Qiang Li**,[1] **Rui-Dong Chen**,[1] **and Xiao-Lei Liu**[3]

[1]*School of Computer Science and Engineering, Institute for Cyber Security, University of Electronic Science and Technology of China (UESTC), Chengdu 611731, China*
[2]*Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518040, China*
[3]*Institute of Computer Application, China Academy of Engineering Physics, Mianyang 621900, China*

Correspondence should be addressed to Xiao-Lei Liu; liuxiaolei@caep.cn

Received 29 October 2020; Revised 19 February 2021; Accepted 14 March 2021; Published 7 April 2021

Academic Editor: Huaizhi Li

APT malware exploits HTTP to establish communication with a C & C server to hide their malicious activities. Thus, HTTP-based APT malware infection can be discovered by analyzing HTTP traffic. Recent methods have been dependent on the extraction of statistical features from HTTP traffic, which is suitable for machine learning. However, the features they extract from the limited HTTP-based APT malware traffic dataset are too simple to detect APT malware with strong randomness insufficiently. In this paper, we propose an innovative approach which could uncover APT malware traffic related to data exfiltration and other suspect APT activities by analyzing the header fields of HTTP traffic. We use the Referer field in the HTTP header to construct a web request graph. Then, we optimize the web request graph by combining URL similarity and redirect reconstruction. We also use a normal uncorrelated request filter to filter the remaining unrelated legitimate requests. We have evaluated the proposed method using 1.48 GB normal HTTP flow from clickminer and 280 MB APT malware HTTP flow from Stratosphere Lab, Contagiodump, and pcapanalysis. The experimental results have shown that the URL-correlation-based APT malware traffic detection method can correctly detect 96.08% APT malware traffic, and its recall rate is 98.87%. We have also conducted experiments to compare our approach against Jiang's method, MalHunter, and BotDet, and the experimental results have confirmed that our detection approach has a better performance, the accuracy of which reached 96.08% and the F1 value increased by more than 5%.

1. Introduction

Advanced Persistent Threats (APT) are the utmost challenging attacks as attackers use sophisticated attacking options to launch persistent attacks on specific targets [1]. With the assistance of APT malware, attackers could remotely control compromised devices and steal high-value information of government, military, and the financial industry. Economic losses of an APT victimized organization amount to millions of dollars [2]. And, the unfortunate thing is that APT attacks are hard to be detected due to their strong concealment, long periods of time, and customization for the targeted organization. APT is generally divided into seven stages: reconnaissance, weaponized deployment, load delivery, exploitation, installation, command and control (C & C), and action [3]. APT attackers' intentions are to steal sensitive information instead of causing damage. Therefore, after the target network is compromised, attackers will install APT malware such as a Trojan horse or backdoor on the infected device to remotely control and steal confidential data for a long period of time. For example, BITTER attack organization used spear-phishing and Microsoft Office-related vulnerabilities to induce a victim to download a malicious Trojan horse [4] to establish communication with C & C, and downloaded various remote control plug-ins according to the command returned by C & C, and then executed a series of malicious actions, such as stealing sensitive data and controlling botnets.

To avoid detection, the most popular APT attackers establish a connection between infected machines with command and control servers through HTTP/HTTPS protocols. It also has contributed to discovering parts of malware traffic that do not rely on HTTPS protocol to build a command and control channel through analyzing HTTP traffic. The rule-based detection method [5, 6] extracts signature rules from a large amount of C & C traffic, and then it generates templates based on these rules to filter normal traffic which does not conform to characteristics of C & C. Despite its promising results on APT malware detection, it cannot identify the normal traffic with similar characteristics of C & C traffic. To solve this problem, supervision-based detection method, which combines the statistical characteristics of HTTP traffic and machine learning, is sought [7]. It mainly trains a classifier based on a large amount of labeled data. In order to decrease the cost of labeling manually, unsupervised methods are proposed to detect APT malware infection, which cluster unlabeled traffic and compare the similarity between suspicious traffic and malicious traffic. However, this method requires a large amount of data for cluster and similarity analysis, resulting in a huge space overhead. To further enhance the performance of detection, correlation-based approach [8] is used. It analyzes the correlation and similarity of malware behaviour and uses different detection methods comprehensively. With the development of malicious attack methods, the concealment of malicious traffic increases, making it increasingly difficult to distinguish normal and malicious traffic, so correlation-based detection method needs improvement.

In this paper, we propose an effective HTTP-based APT malware infection using URL correlation. Different from most existing correlation-based detection methods, we use graph analysis to discover APT malware based on the dynamic correlation of normal traffic. The user's normal web requests have a definite correlation, and the requests generated by APT malware will not be related to the current user's target behaviour. Thus, we can build a web request graph and then identify HTTP-based malware traffic based on the correlation analysis among URLs. Our approach includes three phases: the first phase is to build a web request graph, which is established according to the Referer field in the HTTP header; the second phase is to refine the web request graph through redirection refactoring and URL similarity; and the third step is to filter the remaining unrelated legitimate requests using a normal uncorrelated request filter, which is trained with the user's normal traffic. The remaining outliers in the request graph may be requests initiated by the malware to establish a C & C channel through HTTP protocol. We have conducted experiments on datasets from clickminer [9], Stratosphere Lab [10], Contagiodump [11], and pcapanalysis [12], which demonstrate that our approach has a better performance than Jiang's method [5], MalHunter [13] and BotDet [14].

In summary, our contributions are as follows:

(1) We have proposed an approach to detect HTTP-based APT malware infection based on graph reasoning and used Hviz [15] to construct a web request graph.

(2) Due to a small percentage of normal requests that do not include a *Referer* field, we have proposed two methods, namely, redirection refactoring and URL similarity to add "missing" user-initiated requests into the web request graph and refine the web request graph.

(3) We have used Local Outlier Factor to build a filter, which can filter 83.2% of normal uncorrelated requests.

The rest of this paper is structured as follows: Section 2 introduces the related work of malware C & C channel detection, followed by Section 3, wherein the motivation of using URL correlation analysis to detect APT malware is sketched in detail. Section 4 illustrates our APT malware infection detection based on URL correlation analysis in detail. And, Section 5 presents two methods to refine the web request graph. Section 6 describes the experimental setup and analyzes the experimental results. Discussion and Conclusion sections are summarized in Section 7 and Section 8, respectively.

2. Related Work

Detecting C & C channels is one of the most effective ways to detect APT. Based on the different detection granularity and the basic ideas behind C & C, detection methods are subdivided into rule-based detection, machine-learning-based detection, and correlation-based detection [5].

2.1. Rule-Based C & C Detection. The rule-based detection method mainly proposes detection rules and generates rule templates based on the differences between C & C traffic and normal traffic, and then matches the rule of traffic to be detected with the generated template to determine whether it is malicious C & C traffic. Giroire et al. [16] found that C & C traffic is persistent during access, and persistence is a measure of temporal regularity. Thus, they calculated the "persistence" value of each target atom (the set of target addresses) and considered that atoms with a high persistence score but not on the whitelist were C & C targets. Therefore, C & C traffic communication time is usually used to detect C & C traffic. Sadhan et al. [17] extracted count-feature sequences from the network traffic and then evaluated their periodograms to check for periodic behaviour. These two methods cannot adequately detect variable C & C traffic because they only use a single temporal feature rule to detect. To solve this problem, some works use multiple signature rules to filter non-C & C traffic, and then further analyze the filtering results. Jiang et al. [5] proposed two heuristic rules to filter normal traffic that behaved significantly different from C & C traffic. In order to further improve the detection accuracy, they performed spatial anomaly detection based on spatial characteristics and payload anomaly detection based on statistical characteristics of malicious payloads for the remaining suspicious traffic.

In order to improve the detection speed, the method of extracting traffic characteristics and rules to generate templates and using template matching for C & C traffic detection has appeared. Zarras et al. [18] proposed a botnet detection system, BOTHOUND, which combined HTTP header field chains and HTTP header message templates to identify botnet traffic accurately. Nelms et al. [19] developed ExecScent, which generated a control protocol template (CPT) based on known malicious C & C traffic and matched the measured traffic with the template for similarity. The traffic exceeding the detection threshold would be regarded as C & C traffic.

Due to changes in the network environment and botnet code, not all C & C traffic strictly follows rules and templates. In addition, it is difficult for rule-based detection methods to distinguish normal traffic similar to C & C traffic correctly. In contrast, our method concentrated on the dynamic correlation between normal HTTP traffic without specific rules and used a customized filter to filter normal traffic similar to C & C traffic.

2.2. Machine-Learning-Based C & C Detection. The machine-learning-based detection method extracts features from network traffic, combines machine learning algorithms for model training, and finally uses the trained model to detect C & C traffic. For example, Tegeler et al. [20] presented BotFinder system, which extracted five statistical features from different bot families' C & C traffic and then trained the model using clustering. BotFinder compared the statistical features of the new trace with the trained model's clusters to detect malware. Bilge et al. [21] developed Disclosure to identify C & C servers by extracting flow-size-based features, client access pattern-based features, temporal features, and using a trained detection model based on labeled training samples. To handle encrypted network communication protocols, Zhao et al. [22] came up with a botnet traffic detection approach without depending on the packet payload. They used flow interval behaviour features and decision tree algorithm to discover botnet activity. Mizuno et al. [23] developed a malware-infected device detection system, BotDetector. BotDetector generated a feature template based on the extracted HTTP header fields and adopted classification algorithms to train the detection models. This method only needs to save the HTTP header field, which reduces the amount of information to be kept. Instead of extracting statistical features, this method automatically generates template features based on HTTP headers. Thus, it achieved a high accuracy of classification. In order to mimic the legitimate HTTP communication and hide C & C activities, Sakib et al. [24] proposed an HTTP-based botnet C & C traffic detection method using HTTP request URL field and DNS response packet fields that included two stages. The first stage applied three different anomaly detection methods independently in an unsupervised manner to isolate the software-agent-generated HTTP packets from the browser-generated HTTP packets. The second stage used one anomaly detection method to isolate the botnet C & C domain from the legitimate web domains. To reduce the impact on the detection performance when the C & C server is temporarily offline or changes its response content, Li et al. [13] developed the MalHunter system. MalHunter used a trained classifier to detect malware based on behaviour-related statistical characteristics from HTTP requests. It also used L2 regularization for preventing binary classification model training from overfitting to some degree. Guo et al. [25] applied pronunciation and TLD features into machine learning to improve the accuracy of malicious traffic detection and used statistical methods to reduce the false-positive rate.

Machine-learning-based detection method requires large amounts of labeled dataset for feature engineering and model training. However, our method does not need to use labeled malicious C & C data to train the model. The filter we use only needs to be trained for normal benign traffic. We focused on the correlation between reference URLs and requested URLs from HTTP traffic to overcome the lack of large-scale labeled APT malicious datasets.

2.3. Correlation-Based C & C Detection. Based on the spatial-temporal correlation and similarity of traffic, the correlation-based detection method uses correlation algorithms to perform correlation analysis on network behaviour to detect C & C traffic. BotSniffer [26] was based on their proposed anomaly detection algorithm and performed a group analysis of spatial-temporal correlation and similarity to detect the botnet. Similarly, BotMiner [27] clustered similar communication activities and malicious activities in different planes, and then performed cross-cluster correlation to identify the infected hosts. However, BotSniffer [26] cannot effectively discover C & C activities without centralized servers. In recent years, the correlation-based detection method has improved the accuracy of C & C detection by using the results of multiple detection models to perform correlation detection on suspected C & C traffic. For example, Ghafir et al. [14] used four detection methods to detect C & C attack events, and then used a C & C attack correlation alert framework to perform correlation analysis on C & C attack events to improve the detection accuracy.

The correlation of these methods mainly analyzes the correlation or similarity of malicious traffic characteristics or attack behaviours, or comprehensively considers the results of different detection methods. Unlike these correlation-based detection methods, the correlation of our proposed approach refers to the dynamic association between HTTP request traffic. This association can be represented by the relevant URLs of the web request graph without massive quantities of the dataset. In addition, the dynamic association of normal traffic is difficult to extract, making it difficult for attackers to disguise malicious traffic as normal traffic in our detection method.

3. Motivation and HTTP Request Classification

3.1. Motivation. When a user is browsing the Internet, each web page may include hundreds of embedded objects such as images, videos, or JavaScript code, which results in a lot of

HTTP requests to ask the server for resource. Usually, the initial HTTP request is sent to ask for HTML code, and the corresponding HTTP response message type is the text/html. Then, the browser needs to load JavaScript, CSS, font files, and images referenced by the requested pages based on the HTML code. When the browser is loading the requested page based on the HTML code, users may further trigger the download of other files or AJAX requests, which generates a lot of HTTP requests. Thus, the order of HTTP requests about a web page is related to the user's browsing behaviour and the web page composition. Furthermore, HTTP request packet stores the order of HTTP request, where Host field in Header and Request-URI field in Request line are combined to represent the currently requested page. Referer field in Header represents its previous page. Therefore, the request relationship of each page accessed by the browser can be represented by a Web request graph based on the Referer field.

For example, Figure 1 illustrates the process of constructing a web request graph, where a node represents the requested URL; a directed edge indicates the sequence between two URLs. The page of Chinadaily includes two pictures. In the process of loading the page of Chinadaily, two HTTP requests about these two pictures whose Referer field is the URL of Chinadaily are sent to server. In this case, we add two directed edges from the URL of Chinadaily to URLs of these two pictures in the graph. When a user clicks a link to Page1, the Referer field of the HTTP request for Page1 is the URL of Chinadaily. Then, a directed edge from the URL of Chinadaily to the URL of Page1 is added into the graph. According to HTTP requests issued by Page1, we can add other directed edges into the graph.

In addition to users' legitimate HTTP requests, there are HTTP requests initiated by HTTP-based APT malware to establish communication with the C & C server. Many malicious HTTP requests hide in a lot of users' legitimate HTTP requests to avoid being found by network inspectors. In this work, we found that HTTP requests initiated by HTTP-based APT malware do not contain Referer fields and are not related to users' web browsing behaviour. In other words, HTTP request of APT malware C & C are independent of the web request graph. Therefore, in the process of constructing a normal web request graph, URLs requested by malware cannot be added into the web request graph and become isolated nodes. According to the above characteristics between normal HTTP requests and malicious HTTP requests of HTTP-based APT malware, we propose to use a web request graph based on URL correlation to detect HTTP-based APT malware traffic.

3.2. HTTP Request Classification. In the detection process, many HTTP requests need to be processed. In order to further improve the detection efficiency, we classify and filter HTTP requests generated in the monitored network environment. In this section, we divide HTTP requests into the following four types according to how they are generated.

(1) *User-Initiated Requests*. User-initiated requests are triggered by the behaviours of the user browsing the Internet, including typing an URL into the browser address bar, clicking on a link, browsing a bookmark, or submitting a web form, etc. The Referer field of these requests usually is empty.

(2) *Webpage-Contained Requests*. Webpage-containing requests are mainly JavaScript, CSS, font files, and images referenced during the page loading process, as well as file downloads and AJAX requests automatically triggered by the browser during the user browsing process but not triggered by the user directly. These requests usually include a Referer field that identifies their source URL.

(3) *Normal Uncorrelated Requests*. Normal uncorrelated requests refer to benign users' web requests which lack Referer field. It is generated by redirection or some legitimate software or normal services running on the user device, etc. Without the Referer field, this type of HTTP requests cannot be added into the web request graph directly. Fortunately, normal uncorrelated HTTP requests are related to user normal browsing, and we can add them into the graph or filter them according to rules.

(4) *Malicious Requests*. Malicious requests are generated when APT Malware communicates with C & C servers. These HTTP requests are unrelated to the above three HTTP requests. They do not contain Referer fields and thus cannot be added into the web request graph. Therefore, malicious requests remain uncorrelated HTTP requests.

4. APT Malware Infection Detection Based on URL Correlation Analysis

In this section, we present an overview of the proposed approach for using web request graph based on URL correlation to detect HTTP-based APT malware, as shown in Figure 2. It is mainly divided into four parts: (1) pre-processing, (2) constructing a web request graph, (3) refining the web request graph, and (4) normal uncorrelated requests filtering.

4.1. Preprocessing. The preprocessing stage mainly extracts the relevant URL from HTTP request traffic to facilitate the subsequent construction of the web request graph. We use mitmproxy [28] to capture HTTP traffic from the monitored network and divide them according to users' IP address. Then we extract the Referer field, Host field, and Request-URI field from HTTP request packet and combine Host field and Request-URI field as the request URL. Reference URL from Referer field and request URL are used to construct the web request graph in the next section. HTTP response traffic is used to refine the constructed web request graph in the subsequent section.

4.2. Constructing Web Request Graph. Referring to Hviz [15], we construct a different web request graph according to different users' IP address. In a web request graph, a node

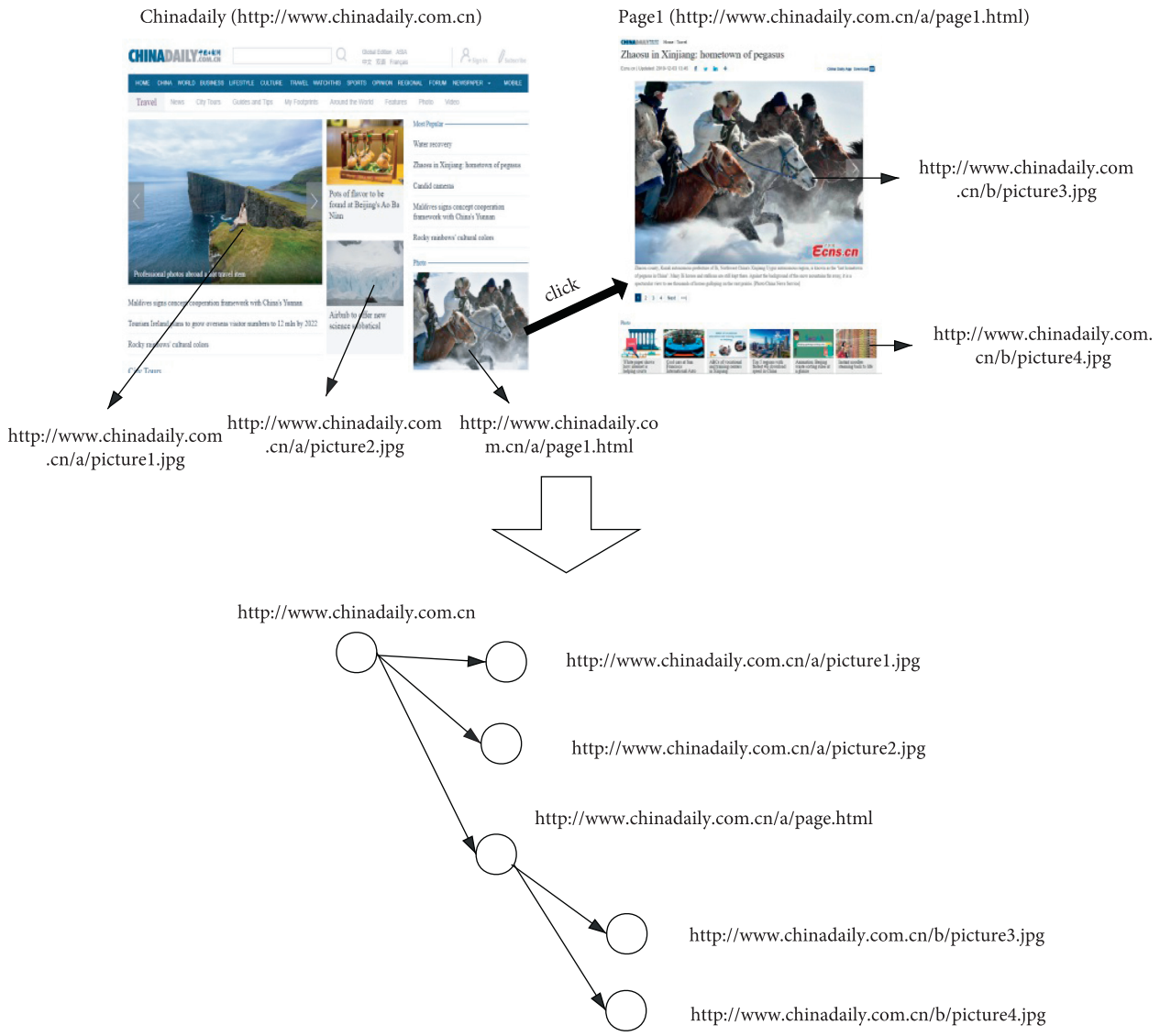

Figure 1: A web request graph based on URL correlation.

which is represented by URL corresponds to an HTTP request and its response. If an HTTP request has a valid Referer field, we add a directed edge from the node corresponding to the reference URL to the node corresponding to the requested URL. For example, if the Referer field of request j is the URL of request i, two nodes i and j can be connected using a directed edge (i, j). In other words, the request j is issued by the request of i. For most HTTP requests, we can extract directed edges with their valid Referer fields and requested URLs. Generally, there is no Referer field in user-initiated requests, and these HTTP requests are the initial nodes of a web request graph.

4.3. Refining the Web Request Graph. Unfortunately, a small percentage of normal requests do not include a Referer field, for example, directly entering the URL address of a resource in the address bar of the browser, visiting the website through the bookmark maintained by the browser, clicking the link in the external application, etc. Therefore, the web request graph constructed based on the Referer field of HTTP requests is incomplete. To refine the constructed web request graph, we proposed to add normal uncorrelated requests into the graph. This section mainly introduces how to add normal uncorrelated requests to the web request graph based on Referer field. According to the generation of normal uncorrelated requests, we come up with two methods to refine the web request graph, namely, redirection refactoring and URL similarity.

4.3.1. Refining the Web Request Graph by Redirection Refactoring. If a user-initiated request is redirected, the Location field of its response will give the redirection address. The browser will automatically jump to the redirection address and send an HTTP request to the redirection address. The subsequent HTTP requests are correlated to the HTTP request of the redirection address. Unfortunately, the

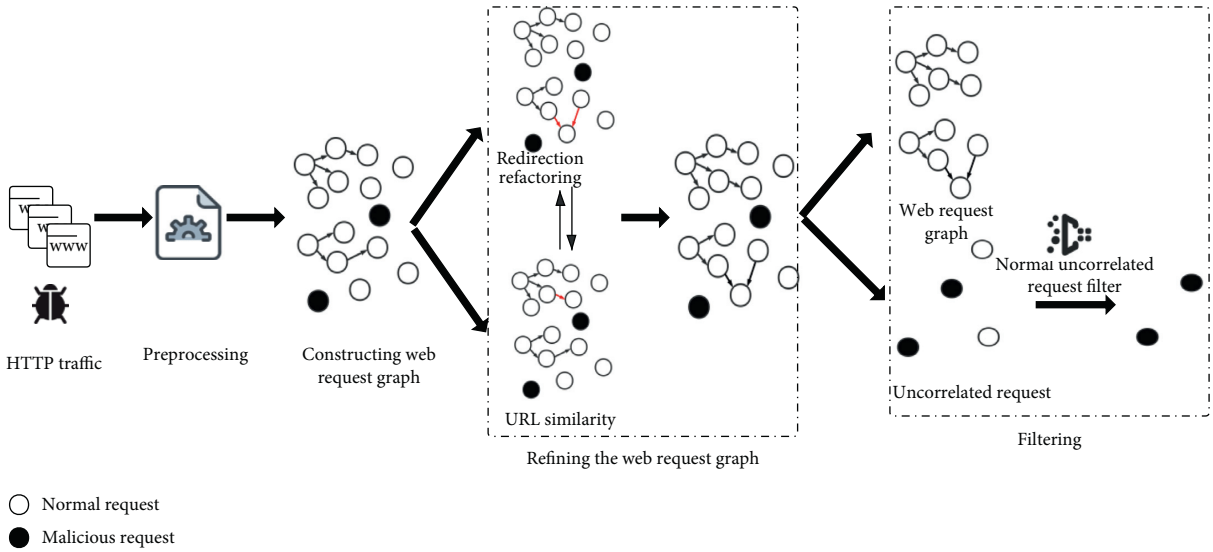

Figure 2: The overview of HTTP-based APT malware traffic detection based on URL correlation.

Referer field of the HTTP request of the redirection address is empty, which means that there is no HTTP request connected to the user-initiated request. Thus, the user-initiated request will become an uncorrelated request. Due to the lack of some user-initiated requests in the web request graph, the accuracy of HTTP-based APT malware detection will be reduced. In the dataset we used, the proportion of redirects was 9.37%. In order to solve the problem, we introduce redirection refactoring to refine the web request graph according to the characteristics of redirection, and add these "missing" user-initiated requests into the web request graph. We will represent the redirection refactoring in detail in part 1 of Section 5.

4.3.2. Refining the Web Request Graph by URL Similarity. In this work, we found that normal HTTP requests with valid Referer fields are only 80% in addition to HTTP requests whose response is redirection. In other words, after redirection refactoring, there are still many normal uncorrelated requests. And, we also found that URLs of normal uncorrelated requests have similarity with other URLs of normal request from the same server. To increase the accuracy of detection, URL similarity is used to add normal uncorrelated requests to the web request graph. The URL similarity focuses on the word-level similarity between URLs. URLs are divided into single word based on the segmentation symbol of URLs and the number of words is counted. Then, we compare the similarity between URLs of normal uncorrelated requests and each URL of the web request graph one by one, and add normal uncorrelated requests into the web request graph. We will introduce URL similarity in detail in part 2 of Section 5.

4.4. Normal Uncorrelated Request Filtering. In addition to users' normal web browsing, there are many legitimate services and legitimate software in the user's device using HTTP protocol to communicate. There are some HTTP requests issued by legitimate software and services. At the same time, these requests cannot be associated with the Web request graph and become isolated nodes. If we do not deal with these requests, they will be misidentified as malicious requests in subsequent detection. To deal with normal requests generated by legitimate services and software, we use machine learning to construct a normal request filter to filter these requests. We use the features referred to the feature selection of the existing machine learning methods shown in Table 1 as input for machine learning, such as TLD, User Agent. We count all the top-level domains in the dataset, and then calculate the probability of each top-level domain appearing in the normal domain name and the malware domain name, and use the ratio of the two probabilities to predict whether the domain name to be tested is a normal domain name. To use the HTTP correctly, the malware needs to create the HTTP header in which the User-Agent field should be filled. A user-agent field in every HTTP request can have one of the following forms: Legitimate user's browser, Empty, Specific, Spoofed, or Discrepant. So, we extract the user-agent field as a feature.

Since the filter is only used to filter normal traffic, we use labeled normal request traffic for training. The selection of machine learning is shown in Section 6.3.

5. Refining the Web Request Graph

5.1. Redirection Refactoring. Redirection refers to redirecting a network request to a new URL which is different from the originally requested URL. After receiving a redirection request, the web server will send its response with a new URL to the user. The client will automatically send a new HTTP request whose URL is the new URL to the server. In general, we can detect whether the HTTP request URL is redirected according to the *Status-Code* of the HTTP response header. If the value of *Status-Code* in the HTTP response header

Table 1: Feature set for normal uncorrelated request filter.

Feature	Description
URL length	Number of characters of the URL
URL entropy	The information entropy of the URL
Number of URL parameters	Number of parameters of the URL
TLD	The top-level domain of the URL
Domain entropy	The information entropy of the domain
Content type	Content type of the HTTP request
Cookie	Does the HTTP request contain cookies?
User agent	User agent of the HTTP request

equals to 301, 302, 303, or 307, it indicates that a redirect has occurred. In addition to setting the status code to redirection, other types of redirection include HTML-based redirection and JavaScript-based redirection. These kinds of redirections are mainly implemented using ⟨meta⟩ element and the *window.location* attribute of the HTML page with a Status-Code of 200. According to the characteristics of page redirection, we replay an HTTP response packet with a Status-Code of 200 to obtain its HTML page. Then, we inspect the *hmetai* element and the *window.location* attribute of the HTML page to determine whether the HTTP request was redirected and obtain the redirected URL. The redirection-based web request graph reconstruction process is shown below.

For example, when one user accesses page B from page A, and if page B is redirected to page C, there is an HTTP response with the address of page B. After the client receives the HTTP response, the client will be sent a new HTTP request to visit page C. Actual redirection edges of the above situation are $A \longrightarrow B$, $A \longrightarrow C$, and subsequent requests are connected to node C in the web request graph, as shown in Figure 3(a). To refine the web request graph, our proposed method will reconstruct these direction edges as $A \longrightarrow B \longrightarrow C$, which is shown in Figure 3(c). The HTTP request of page C occurs after that of page B, thus we delete the directed edge from node A to node C and add the directed edge from node B to node C, as shown in Figure 3(b). If a user directly requests page B, the Referer field of his or her HTTP request is empty. Because it is redirected to page C, subsequent HTTP requests are connected to page C, then the HTTP request for page B becomes an outlier in the web request graph. However, the http request for page B is a normal uncorrelated request, as shown in Figure 4(a). In order to improve the detection accuracy, we restructure the web request graph: adding the HTTP request of page B to the web request graph, that is, letting node B connect to node C, as shown in Figure 4(b). The reconstructed result is shown in Figure 4(c).

5.2. URL Similarity. URL is a uniform resource locator, which can indicate the location of resources in the server. Different URLs that request resources with the same scope or the same type of resources on the same server may have similarities. Some URLs of normal uncorrelated requests similar to the web request graph, share the same parent URL. Based on the above characteristics, we can calculate the similarity between the URL of uncorrelated requests and URLs of the web request graph, and add these normal uncorrelated requests to the web request graph based on URL similarity. There are several calculation methods, like string similarity, space vector similarity, string edit distance, and clustering-based method. Considering the principles and applicability, we choose a similarity calculation method for domain name characteristics. The URL similarity calculation process is as follows:

According to the characteristics of the URL, using "." and "/" to divide the resource access level, we propose to hierarchically divide each URL into a set of elements. Each URL is divided into two parts based on the Host and Request-URI fields. The host field is split by "." while the Request-URI field is split by "/," and these split elements make up a collection of URL elements. After splitting the URLs that access the same server, it is found that these URLs have the same elements. Therefore, we proposed to count the number of elements of URLs, and calculate the ratio between the number of same elements and the maximum number of two URLs. Therefore, we came up with the following equation. Counting the number of elements in a URL collection and using the following equation, we get the similarity value of two URLs.

$$\alpha = \frac{\gamma(A, B)}{max(\beta(A), \beta(B))}. \quad (1)$$

In this equation, $\gamma(A, B)$ is the number of elements in common of request URLs A and B. $\beta(A), \beta(B)$ is the number of elements of request URLs A and URL B, respectively. $\max(\beta(A), \beta(B))$ refers to the maximum between $\beta(A)$ and $\beta(B)$. For example, there are request URL A (http://www.example.com/test/123.html) and request URL B (example.com/test/abc.html), and the common elements between URL A and B are *example*, *com*, and *test*, which means that $\gamma(A, B) = 3$. Due to $\beta(A) = 5, \beta(B) = 4, \max(\beta(A), \beta(B)) = 5$, the similarity rate α of request URLs A and B is 0.6.

We consider the similarity calculation between an unrelated request URL and all URLs in the web request graph as a URL similarity calculation process. After the similarity calculation process of an unrelated request ends, the maximum similarity value is extracted. If the maximum similarity value is greater than the similarity threshold v, the URL of an unrelated request finds its similar peer URL in the web request graph. This uncorrelated request URL shares the same parent URL as its similar sibling URLs in the web request graph. Its parent URL is then connected to the URL of this unrelated request. URLs of unrelated requests can be successfully added to the web request graph in this way.

6. Experiment and Evaluation

This section mainly introduces the evaluation criteria, datasets, and experimental results analysis.

6.1. Evaluation Criteria. This work will use accuracy, recall, and F1-score to comprehensively evaluate the detection effect.

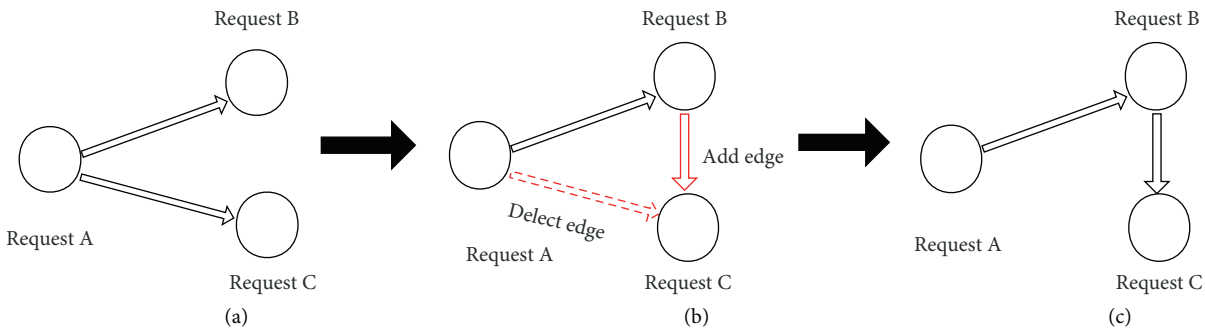

Figure 3: Redirection refactor of normal redirection.

6.1.1. True Positive (TP). TP represents the number of normal HTTP requests, which is predicted as normal, that is, the number of normal HTTP requests that constitute a web request graph.

6.1.2. True Negative (TN). TN represents the number of malicious HTTP requests, which is predicted as malicious, that is, the number of malicious HTTP requests.

6.1.3. False Positive (FP). FP represents the number of malicious HTTP requests, which is predicted as normal, that is, the number of malicious HTTP requests that constitute a web request graph.

6.1.4. False Negative (FN). FN represents the number of normal HTTP requests, which is predicted as malicious, that is, the number of remaining normal uncorrelated HTTP requests.

6.1.5. Accuracy. Accuracy is the ratio of the number of correctly detected HTTP requests to the number of HTTP requests to be detected. The higher the accuracy, the better the detection. Accuracy calculation is as follows:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{2}$$

6.1.6. Recall. Recall is a measure of coverage. Recall is the ratio of correctly detected normal HTTP requests to the number of actual normal HTTP requests. The higher the recall rate, the better the detection. Recall rate is calculated as follows:

$$\text{recall} = \frac{TP}{TP + FN}. \tag{3}$$

6.1.7. F1-Score. The definition of F1-score is based on precision and recall. The higher the F1-score, the better the detection. F1-score is calculated as follows:

$$\text{precision} = \frac{TP}{TP + FP}, \tag{4}$$

$$F1 - \text{score} = \frac{2^{*}\text{precision}^{*}\text{recall}}{\text{precision} + \text{recall}}. \tag{5}$$

6.2. Dataset. The experimental dataset used in this work includes the normal traffic dataset and malicious traffic dataset. We used a normal dataset from clickminer [9], which contains 24 traces. These traces were collected from a user study with 21 participants from the University of Georgia. All participants were assigned a browsing time slot of about 20 minutes, during which they visited a large variety of sites. The total size of the dataset is 1.48 GB, with more than 70,000 HTTP requests. The http request generated by the software is consistent with that generated by the browser, so the dataset can be used to represent the traffic generated by the software and the browser. Among the normal traffic dataset, we use 1 GB traffic to train a normal uncorrelated request filter, and the rest is used to evaluate our experiments. The other part of the dataset is the malicious dataset. The malicious traffic datasets were derived from APT malware collected by Stratosphere Lab [10], Contagiodump [11], and Pcapanalysis [12]. In addition, the size of the malicious traffic dataset we used is 280 MB with over 10,000 HTTP requests.

6.3. Selecting Parameters. Before implementing the experiments, we need to choose a suitable machine learning algorithm to build a normal request filter. Since only normal request traffic is used to train, malicious traffic is an outlier for the normal request filter. Based on such characteristics, we use the anomaly detection algorithm to train a normal uncorrelated request filter, including One Class SVM, Isolation Forest, and Local outlier Factor. The training data are mainly constructed by normal uncorrelated request traffic, which is the remaining request traffic of clickminer after constructing the web request graph. The experimental results are shown in Table 2. In our experiment, when the Local Outlier Factor is selected and set n_neighbors = 7, the normal uncorrected filter performs better and its accuracy reaches 92.91%.

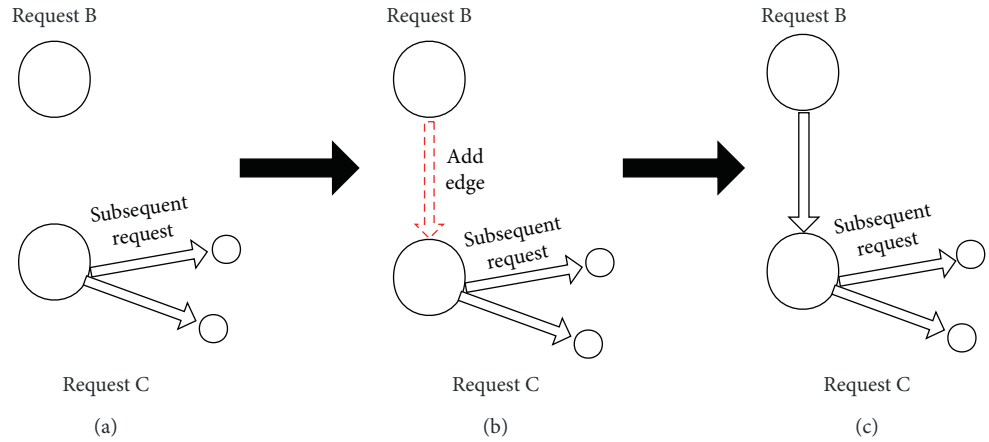

Figure 4: Redirection refactor of user directly request.

Table 2: Result of different machine learning.

Machine learning	Accuracy (%)	Recall (%)	$F1$-score (%)
One class SVM	90.49	91.69	93.87
Isolation forest	89.81	92.87	93.50
Local outlier factor	92.91	95.31	95.48

Table 3: Change of uncorrelated requests during experiment.

Experiment step	Normal (%)	Malicious (%)
Construct web request graphs	6.02	19.84
Refine web request graphs	5.20	19.47
Normal uncorrelated request filter	0.87	19.01

6.4. Experiment Results and Analysis. After splitting the URLs that access the same server, it is found that these URLs have the same elements. We found that when the URL similarity threshold ν is 0.5, the detection effect is better. Since most URLs have the same top-level domain name (.com, .edu, etc.) or server name (such as www), if the threshold selected is too small, many unrelated URL nodes will be added into the web request graph. If the threshold is chosen too large, the most legitimate unrelated URL nodes cannot be added into the request graph.

We mixed the collected dataset and then performed experiments in a computing environment with 3.6 GHz Intel Core i5 and 8 GB of RAM.

As shown in Table 3 and Figure 5, using the proposed method in this paper, the number of normal uncorrelated requests was significantly reduced, and the accuracy of detection increased. After extracting the web request graph from HTTP traffic, there were still 6.02% normal uncorrelated requests. By refining the web request graph, the normal uncorrelated requests were reduced by 13.62% and the accuracy was 91.75%. After filtering the normal uncorrelated requests, the normal uncorrelated requests were reduced by 83.20%, and the accuracy of detecting APT malware traffic was 96.08%. Clearly, refining the web request graph and normal uncorrelated request filtering effectively reduced the normal uncorrelated requests and effectively improved the effect of APT malware traffic detection. However, in the process of refining the web request graph, the malicious requests were reduced by 1.84%. We analyzed this situation in detail and found that a small part of malicious requests were added into the web request graph due to URL similarity. The main reason is that the malicious dataset contains some legitimate traffic, but it was wrongly labeled as malicious.

6.5. Comparative Experimental Results Analysis. We compared our proposed detection method with three types of detection methods, which are Jiang's [5] method of rule-based detection, MalHunter [13] of machine-learning-based detection, and BotDet of correlation-based detection. Jiang' method mainly used two heuristic rules to filter most non-C & C traffic, and then used two anomaly detection models to filter normal traffic, which is similar to C & C traffic. MalHunter focused on header features of malware traffic. MalHunter extracted URL features, HTTP header features, and HTTP header sequence features, and then adopted XGBoost to detect malware traffic. BotDet proposed a C & C correlation alert framework, which correlated four method detection results, making C & C traffic detection more accurate.

As for the reference [8], similar with our work, they use click detection, whitelist filtering, and links to complete the picture. However, click detection has limitations. Because it depends on the training data, the accuracy of the model will decrease in a real network environment and lead to a decrease in the effect of graph reconstruction. At the same time, whitelist filtering has a certain lag. However, we train normal benign data by using machine learning in the last step. These data do not depend on a specific dataset and will perform better in the actual network environment.

The results of the comparison experiments were shown in Figure 6. Our proposed method had the best detection effect, the accuracy of which reached 96.08% and the recall reached 98.87%. While BotDet had poorer detection effect, its accuracy was 87.14% and recall was 85.71%. As shown in Figure 7, these four APT malware traffic detection methods have good performance based on the roc curve. However, the area enclosed by our proposed method was larger than the others. These experimental results showed that our

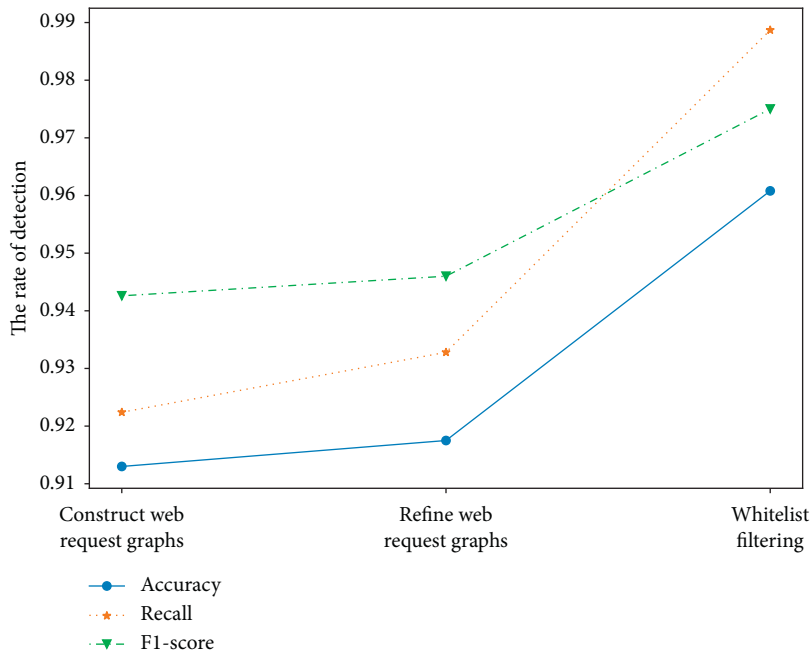

Figure 5: Change of the detection rate during experiment.

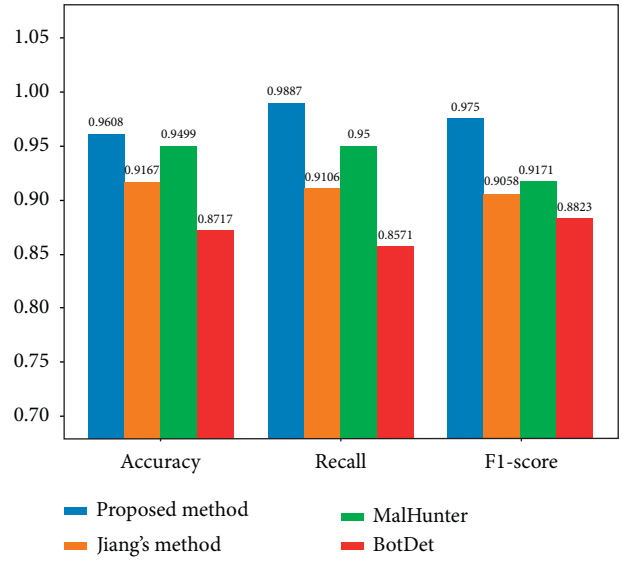

Figure 6: Comparison of our method with three types of detection method.

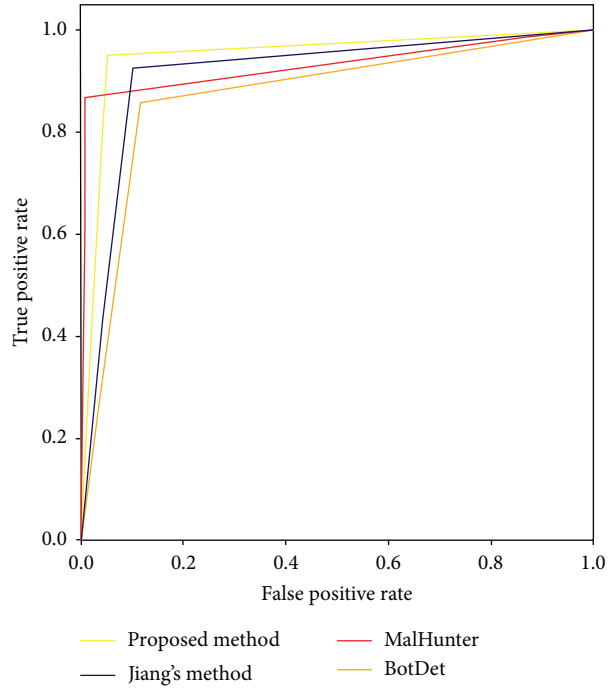

Figure 7: The roc of our method and three types of detection methods.

proposed method was superior to APT malware detection methods of compared experiments.

7. Discussion

Our proposed method also has some limitations. Our main goal was to detect HTTP communications generated by malware whose Referer field is empty. This is because the malware mainly conducts C & C communication with the server, so most of the HTTP traffic Referers generated by the APT malware is empty and become an isolated node in the request graph. If malware forges the Referer field, it will have a certain impact on our method. For example, some malicious software fakes the Referer field as a known site, and may mistakenly judge it as a benign node during the correlation detection process. In our follow-up work, we will continue to solve the problems caused by forged Referer fields. At the same time, the proposed method also has certain detection capabilities for botnets. This is because

when the botnet uses HTTP C & C communication, it will also become an isolated node in the request graph and be detected. For example, Chen et al. [29] proposed that mobile botnets use cloud services to achieve robustness and concealment, but their push services use bi-direction communication channels to transmit information, and they may still be detected. As for the detection of network secret channels, our method is not applicable. This is because the detection method of the converted channel mainly focuses on the content and behaviour of the data packet. For example, Luo et al. [30, 31] proposed a combined method to encode the message into a data packet and the time or content of the request message. Later, they further proposed a method of using time-series channels to transmit hidden information. Our method mainly focuses on the correlation analysis of HTTP requests. The detection of secret channels is a behavioural detection that pays more attention to packet transmission. This will be the direction we will focus on in the next work plan.

8. Conclusion

When HTTP-based APT malware performs C & C communication, it will generate HTTP traffic that is unrelated to the normal HTTP traffic of users' normal Internet browsing. Based on this observation, this paper proposed a method of detecting HTTP-based APT malware-infected traffic using URL correlation. First, we referred to Hviz to construct a web request graph according to URL correlation. Then, we used the redirection refactoring and URL similarity to refine the web request graph. Finally, we used a normal uncorrelated request filter to increase the detection accuracy. After these three steps were completed, the remaining uncorrelated requests were marked as malicious traffic. This paper used public datasets from clickminer, Stratosphere Lab, Contagiodump, and pcapanalysis to evaluate the proposed method. The accuracy of detecting HTTP-based APT malware traffic was 96.08%. The experimental results have shown that the proposed method can effectively detect APT malware traffic. And, the proposed method does not need to rely on the knowledge of previous attacks, and it is more advantageous for detecting small-scale datasets generated by HTTP-based APT malware.

Data Availability

The malicious traffic datasets were derived from APT malware collected by Stratosphere Lab [10], Contagiodump [11], and Pcapanalysis [12]. In addition, the size of the malicious traffic dataset we used is 280 MB with over 10,000 HTTP requests.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partially supported by the National Key Research and Development Program of China (Grant no. 2016QY13Z2302), the National Natural Science Foundation of China (Grant no. 61902262), the National Defense Innovation Special Zone Program of Science and Technology (Grant no. JG2019055), the Director of Computer Application Research Institute Foundation (Grant no. SJ2020A08), and China Academy of Engineering Physics Innovation and Development Fund Cultivation Project (Grant no. PY20210160).

References

[1] M. J. Kim, S. Dey, and S.-W. Lee, "Ontology-driven security requirements recommendation for apt attack," in *Proceedings of the 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, pp. 150–156, IEEE, Jeju, Korea, September 2019.

[2] A. Zimba, H. Chen, Z. Wang, and M. Chishimba, "Modeling and detection of the multi-stages of advanced persistent threats attacks based on semi-supervised learning and complex networks characteristics," *Future Generation Computer Systems*, vol. 106, pp. 501–517, 2020.

[3] 2019, https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.htmls Lockheedmartin.

[4] C. C. San and M. M. Su Thwin, "Selecting prominent api calls and labeling malicious samples for effective malware family classification," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 17, no. 5, 2019.

[5] J. Jiang, Q. Yin, Z. Shi, M. Li, and B. Lv, "A new C & C channel detection framework using heuristic rule and transfer learning," in *Proceedings of the 2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, pp. 1–9, IEEE, London, UK, October 2019.

[6] R. Zhang, Y. Huo, J. Liu, and F. Weng, "Constructing apt attack scenarios based on intrusion kill chain and fuzzy clustering," *Security and Communication Networks*, vol. 2017, Article ID 7536381, 2017.

[7] J. Gardiner and S. Nagaraja, "On the security of machine learning in malware C & C detection," *ACM Computing Surveys*, vol. 49, no. 3, pp. 1–39, 2016.

[8] P. Lamprakis, R. Dargenio, D. Gugelmann, V. Lenders, M. Happe, and L. Vanbever, "Unsupervised detection of apt C &C channels using web request graphs," in *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 366–387, Springer, Bonn, Germany, July 2017.

[9] C. Neasbitt, R. Perdisci, Li Kang, and N. Terry, "Clickminer: towards forensic reconstruction of user-browser interactions from network traces," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1244–1255, ACM, Scottsdale, AZ, USA, November 2014.

[10] Czech Republic, "Stratosphere Lab," 2013, https://www.stratosphereips.org/.

[11] Milaparkour, "Contagio malware dump," 2019, http://contagiodump.blogspot.com.

[12] "Pcap analysis.com," 2019, http://www.pcapanalysis.com.

[13] Ke Li, R. Chen, L. Gu, C. Liu, and J. Yin, "A method based on statistical characteristics for detection malware requests in network traffic," in *Proceedings of the 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pp. 527–532, IEEE, Guangzhou, China, June 2018.

[14] I. Ghafir, V. Prenosil, M. Hammoudeh et al., "Botdet: a system for real time botnet command and control traffic detection," *IEEE Access*, vol. 6, pp. 38947–38958, 2018.

[15] D. Gugelmann, F. Gasser, B. Ager, and V. Lenders, "Hviz: http (s) traffic aggregation and visualization for network forensics," *Digital Investigation*, vol. 12, pp. S1–S11, 2015.

[16] F. Giroire, J. Chandrashekar, N. Taft, E. Schooler, and D. Papagiannaki, "Exploiting temporal persistence to detect covert botnet channels," in *Proceedings of the International Workshop on Recent Advances in Intrusion Detection*, pp. 326–345, Springer, Saint-Malo, France, September 2009.

[17] B. AsSadhan, M. F. José, D. Lapsley, C. Jones, and W. T. Strayer, "Detecting botnets using command and control traffic," in *Proceedings of the 2009 Eighth IEEE International Symposium on Network Computing and Applications*, pp. 156–162, IEEE, Cambridge, MA, USA, July 2009.

[18] A. Zarras, A. Papadogiannakis, R. Gawlik, and T. Holz, "Automated generation of models for fast and precise detection of http-based malware," in *Proceedings of the 2014 Twelfth Annual International Conference on Privacy, Security and Trust*, pp. 249–256, IEEE, Toronto, Canada, July 2014.

[19] N. Terry, R. Perdisci, and M. Ahamad, "Execscent: mining for new c&c domains in live networks with adaptive control protocol templates," in *Proceedings of the Presented as Part of the 22nd {USENIX} Security Symposium ({USENIX} Security 13)*, pp. 589–604, Washington, DC, USA, August 2013.

[20] F. Tegeler, X. Fu, G. Vigna, and C. Kruegel, "Botfinder: finding bots in network traffic without deep packet inspection," in *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, pp. 349–360, ACM, Florham Park, NJ, USA, September 2012.

[21] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "Disclosure: detecting botnet command and control servers through large-scale netflow analysis," in *Proceedings of the 28th Annual Computer Security Applications Conference*, pp. 129–138, ACM, Orlando, FL, USA, December 2012.

[22] D. Zhao, I. Traore, B. Sayed et al., "Botnet detection based on traffic behavior analysis and flow intervals," *Computers & Security*, vol. 39, pp. 2–16, 2013.

[23] S. Mizuno, M. Hatada, T. Mori, and S. Goto, "Botdetector: a robust and scalable approach toward detecting malware-infected devices," in *Proceedings of the 2017 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, Paris, France, May 2017.

[24] M. N. Sakib and C.-T. Huang, "Using anomaly detection based techniques to detect http-based botnet C & C traffic," in *Proceedings of the 2016 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, Kuala Lumpur, Malaysia, May 2016.

[25] Z. Guo, P. Jin, J. Fu, Y. Cheng, and C. Chen, "Botnet detection method based on artificial intelligence," in *Proceedings of the 2019 IEEE Fourth International Conference on Data Science in Cyberspace (DSC)*, pp. 487–494, IEEE, Hangzhou, China, June 2019.

[26] G. Gu, J. Zhang, and W. Lee, "Botsniffer: detecting botnet command and control channels in network traffic," in *Proceedings of the Network and Distributed System Security Symposium*, San Diego, CA, USA, September 2008.

[27] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: clustering analysis of network traffic for protocol-and structure-independent botnet detection," in *Proceedings of the 17th USENIX Security Symposium*, San Jose, CA, USA, August 2008.

[28] "Maximilianhils cortesi. mitmproxy," 2018, https://mitmproxy.org.

[29] W. Chen, X. Luo, C. Yin, B. Xiao, M. H. Au, and Y. Tang, "CloudBot: advanced mobile botnets using ubiquitous cloud technologies," *Pervasive and Mobile Computing*, vol. 41, pp. 270–285, 2017.

[30] X. Luo, E. W. W. Chan, P. Zhou, and R. K. C. Chang, "Robust network covert communications based on TCP and enumerative combinatorics," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 6, pp. 890–902, 2012.

[31] X. Luo, P. Zhou, E. W. W. Chan, R. K. C. Chang, and W. Lee, "A combinatorial approach to network covert communications with applications in Web leaks," in *Proceedings of the 2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN)*, pp. 474–485, Hong Kong, China, June 2011.