Hindawi
Security and Communication Networks
Volume 2020, Article ID 8841544, 20 pages
https://doi.org/10.1155/2020/8841544

Research Article

MD-MinerP: Interaction Profiling Bipartite Graph Mining for Malware-Control Domain Detection

Tzung-Han Jeng,[1,2] **Yi-Ming Chen**,[2] **Chien-Chih Chen**,[1] **and Chuan-Chiang Huang**[1]

[1]*Chunghwa Telecommunication Laboratories, Taoyuan, Taiwan*
[2]*National Central University, Taoyuan, Taiwan*

Correspondence should be addressed to Tzung-Han Jeng; tzunghan@cht.com.tw

Received 9 April 2020; Revised 10 July 2020; Accepted 16 October 2020; Published 29 October 2020

Academic Editor: Hammad Afzal

Despite the efforts of information security experts, cybercrimes are still emerging at an alarming rate. Among the tools used by cybercriminals, malicious domains are indispensable and harm from the Internet has become a global problem. Malicious domains play an important role from SPAM and Cross-Site Scripting (XSS) threats to Botnet and Advanced Persistent Threat (APT) attacks at large scales. To ensure there is not a single point of failure or to prevent their detection and blocking, malware authors have employed domain generation algorithms (DGAs) and domain-flux techniques to generate a large number of domain names for malicious servers. As a result, malicious servers are difficult to detect and remove. Furthermore, the clues of cybercrime are stored in network traffic logs, but analyzing long-term big network traffic data is a challenge. To adapt the technology of cybercrimes and automatically detect unknown malicious threats, we previously proposed a system called *MD-Miner*. To improve its efficiency and accuracy, we propose the *MD-Miner*^{P} here, which generates more features with identification capabilities in the feature extraction stage. Moreover, *MD-Miner*^{P} adapts interaction profiling bipartite graphs instead of annotated bipartite graphs. The experimental results show that *MD-Miner*^{P} has better area under curve (AUC) results and found new malicious domains that could not be recognized by other threat intelligence systems. The *MD-Miner*^{P} exhibits both scalability and applicability, which has been experimentally validated on actual enterprise network traffic.

1. Introduction

Cybercrimes are becoming increasingly serious with the proliferation of Internet devices and applications. One of the most frequently used tools for cybercrimes is malicious domains to perform phishing, XSS, and other attacks. Internet attack organizations generally use code obfuscation techniques to generate a large number of polymorphic variants with the same malware [1] before establishing more than one command and control (C&C) server. Cybercriminals and malware authors leverage not only hidden and slow APT attacks but also various techniques, such as DGAs and domain-flux, to make them successful. By adopting technologies such as DGAs, these servers change their domain names and corresponding IP addresses over time to prevent being blocked by antivirus software or intrusion prevention systems [2]. The detection of malicious domains is difficult because of the defense dilemma caused by the long-term attack and the volatility of their domain names. However, malware generally exhibit footprints that show where they have been. The clue to tracking cybercrimes is in the network traffic; the challenge is how to analyze the huge amount of network traffic. Of the applications in malicious domains, botnets are considered the most damaging by enterprises.

A set of infected and controlled entities can be viewed as a botnet [3]. The botnet structure is composed of three main components: (1) the bots, (2) the command and control servers (C&C), and (3) the threat actor, or bot herder itself; bot, which refers to a remote victim computer, usually without the victim's knowledge; and C&C server, responsible for managing the trunk host that controls the entire botnet and passes along the bot herder's instructions. Once the botnet deployment is complete and launches a cyber-

attack, the distributed denial-of-service (DDoS) shuts down the victim organization's Internet service, and the APT leads to additional damage. Compromised hosts need the Internet as a communication bridge to perform cybercrimes, such as receiving instructions or stealing sensitive data and returning it to the C&C servers [4, 5]. The impact of botnets is great enough that several studies have focused their attention on the discovery of botnets, which has continued to be a hot topic [6–10].

To defend against cyber-attacks, many organizations have established systems such as intrusion detection systems (IDS) to detect and log suspicious traffic, but these produce many false alarms that dull their vigilance [11]. Unlike advanced traffic analysis techniques that require large amounts of computational resources and time, the domain blacklist matching method can instantly detect malicious domains and further disrupt their communications. However, the methods to perform string changes to domain names are simple, cheap, and fast, indicating that using domain blacklists to prevent attacks is effective but difficult to update in real time. Therefore, automating the maintenance of the domain blacklist is indispensable to improve the information security of organizations.

As described in our previous research [12], discovering botnets is important, and detecting C&C servers is vital to analyze APT events. Malicious domain names commonly require an Internet connection to communicate with compromised hosts, but tracking or mining them from the global public Internet has been a difficult problem. Fortunately, such processes leave footprints, and most enterprises leverage proxy servers as intermediate HTTP communications between internal computers and the Internet that result in logging footprints. Thus, systems can take advantage of packet capturing systems to obtain the HTTP communication records. However, one of the bottlenecks in analyzing network traffic is a single workstation can easily have millions of packets each day, which inhibits manually analyzing such traffic without automated intelligence systems. Therefore, we proposed the *MD-Miner* (*MD* stands for malicious domain) that adapts big data analysis with a scalability framework. The process utilizes network traffic to build a *Process*-domain annotated graph that discovers who is connecting with what. The *MD-Miner* uses user-agent plus client-IP as a feature to distinguish the distinct processes and incorporates this into the annotated bipartite graph to become the *Process*-domain annotated graph. The evaluation in [12] shows that the *MD-Miner* can determine a part of unknown domains that has a high probability of being malicious and demonstrates great identifiability, but there is still room for further improvement.

Inheriting from our previous research [12], we built a new scalable network-level behavior system called *MD-Miner*^{P} (^{P} represents Plus) that is based on the Hadoop and Spark cluster architecture. The design effectively uses an incremental clustering algorithm to handle large amounts of data. The *MD-Miner*^{P} has evolved unique analytic capabilities that constantly examine the subtle clues left in proxy or network traffic logs to discriminate malicious domains.

This article demonstrates the steps to convert the *MD-Miner* to the *MD-Miner*^{P} through two key points. First, the *MD-Miner*^{P} replaces the annotated bipartite graph with an interaction profiling bipartite graph that better represents the association of Internet interactions. Second, the *MD-Miner*^{P} exploits more connection factors to construct features with classification capabilities. In addition to the user-agent plus client-IP (*Process*), the *MD-Miner*^{P} uses HTTP requests, domain IP addresses, and domain name lexical characteristics. The *MD-Miner*^{P} leverages the user-agent plus client-IP building *Process*-domain interaction profiling graph to acquaint process queries that leverage HTTP requests to build the *Trace*-domain interaction profiling graph and determine the interactions between the client-server. The system also leverages the IP address of the destination domain to build the IP-domain interaction profiling graph to identify corresponding relations of the IP used by the domain name. The lexical algorithm is also used to extract variations in the domain string. Finally, these features are aggregated to frame the malicious domain detector. Related works and observations related to improvements of the *MD-Miner*^{P} are detailed in Section 2.

The evaluation stage in Section 4 uses the CyberGraph [13] to verify new malicious domains found by the *MD-Miner*^{P} in addition to the previously used K-fold cross-validation. The CyberGraph is a novel potential malicious domain verification analysis platform that retrieves different types of observable intelligence from different sources to produce a series of observations over time. This allows users to judge threats on the Internet. The CyberGraph is committed to integrating standardized and structured information through a vast and complex network intelligence.

The remainder of this paper is organized as follows. Section 2 describes the background and the assumptions and observations of our approach. Section 3 provides implementation detail of *MD-Miner*^{P} and formulates the research contribution. Moreover, our design goals and core concepts are introduced and a simple example is used to illustrate the data flow of the framework. Section 4 shows the results from our evaluation using ISP-confirmed real-world network traffic to determine the effectiveness of the proposed system. Finally, a summary of the contributions and future research developments are presented in Section 5.

2. Background and Related Work

The principles of related techniques used by the *MD-Miner*^{P} to generate the domain features are described in this section. The *MD-Miner*^{P} has two major evolutions: improvements to the annotated bipartite graph and additional significant features. There are different annotated bipartite graphs imported for feature extraction. In [14, 15], two systems called *Segugio* and *Doctrina* are built from different annotated bipartite graphs with the DNS logs. These systems extract DNS answer-based features, time-based features, domain name-based features, and TTL value-based features of the DNS traffic to detect malicious domain activities. We used annotated bipartite graphs to develop a system, called *MD-Miner*, that monitors the network traffic to build a

Process-domain annotated graph, as shown in Figure 1, to represent who is connecting to what [12]. The *MD-Miner* has abundant DNS logs available and is a scalable architecture. As shown in Figure 1, there are only malicious, benign, and unknown labels in the annotated bipartite graph, but the content of the network traffic log is not as simple as the DNS log. Therefore, the *MD-Miner*^{P} replaces the annotated bipartite graph with an interaction profiling bipartite graph, which is detailed in Section 3 and has experimental results that show promise for its application.

2.1. User-Agent. The first factor in the network traffic log is the user-agent in the HTTP header sent along with a request for an Internet server, which is often but not always sent from a web browser. The intent is to inform the server of the capabilities of the software used by the client. The implementation of a classifier for user-agent strings with support vector machines is described in [16]. On the other hand, as mentioned in [12], the text area of a binary-analyzed result for malware suggests that when the user-agent string is hard-coded in the malware's text area, the user-agent and malicious activities have a considerable degree of correlation. Anomalous user-agent strings were considered in [17] to determine the association with malware activities. However, dedicated user-agent strings that define attackers can easily evade detection by changing their form. Therefore, the *MD-Miner* [12] proposed a *Process*-domain annotated graph that uses user-agent strings and the client-IP in the network traffic as a feature to differentiate the network activity that was emitted from the same process and stores the information about who is connecting to what. In this annotated bipartite graph, the nodes represent either the *Process* nodes (p_1~p_4) or domain nodes (d_1~d_5), and an edge connects a *Process* to a domain if the connection occurred during the considered traffic observation time window. The classification results are used here to construct more effective bipartite graphs based on its composition using the factors described below.

2.2. HTTP Request. The HTTP network traffic contains significant important information to detect malicious interactions between malware-controlled domains and malware-compromised machines. HTTP is an application layer protocol that uses headers to transfer metadata over a client-server model where the client sends a request to a server, which responds with the available appropriate resource. The HTTP requests are important in Internet interactions, making this the second factor used to extract domain features. Many works have confirmed that the vast majority of malware leverages HTTP as a communication bridge with a cybercriminal's C&C server to perpetrate malicious activities [18, 19]. Such tricks are not only used in the majority of SPAM botnets but also operated on the APT [20–25]. In addition, the malware sample network activity experiments in [26] indicate that approximately 75% of malware samples trigger network activities and generate HTTP traffic. A malware clustering system was introduced in [26] to analyze the structural similarities between malicious HTTP requests in network traffic and used the application path and query string to calculate the distance between malware to clustering malware to obtain its signature. In addition, the HTTP request contained in the headers includes the path (e.g., /path/data) and query (e.g., ?key = value&key2 = value2) as ensconced interactive information between the client and server. References [27, 28] tried to detect malicious phishing web sites using path and query keywords by comparing the relevancy of terms within their URLs. One risk level is the similarity between the path and query terms based on Google Trend and Yahoo Clue. In studies that use HTTP protocols to detect suspicious packets [29, 30], the similarity from the URL path, parameter, and value could identify the packet as malicious or benign.

The *MD-Miner*^{P} refers to the *Trace-Channel* interaction profiling graph proposed by our previous research on the *CC-Tracker* [19], which extends similar observations to [26]. The observation is that different malware samples that rely on the same web server application have similarly structured queries and related URL sequences. To reduce the complexity of the computing similarity between HTTP requests, we simplified the HTTP request as *Trace*, as shown in Figure 2. The upper part of Figure 2 shows that the *Trace* takes a raw HTTP request of "GET /web page.php? key1 = value1&key2 = value2&k3 = v3" as an example, where *m* indicates the method to query the URL and *p* denotes the queried page. The remaining terms used to query the URL are *n* and *v*, which are after the question mark and are in the form of a key = value pair, where *n* indicates the parameter name of the queried URL and *v* denotes the parameter. As the parameter values are relatively easy to change, all parameter values are replaced with the same symbol, which ignores the parameter values [19]. Therefore, the original HTTP request can be simplified to "GET_/web page.php?| key1|key2|k3|," as shown in the lower part of Figure 2.

2.3. IP Address. The Internet protocol (IP) address is a unique logical digital address assigned to each hardware-equipped network and is recognized by the other devices through the IP address. Benign and malicious domains also have their own IP address and the correspondences are recorded in the network traffic files. The IP addresses are more stable than other metrics, such as the URL and DNS. That is, the domain string can easily change while the IP address is generally fixed. Cybercrimes create a specific technique called obfuscation to change the domain name string, which has been identified and summarized as having four basic types [31]. In contrast, the IP address holds two inborn traits that make it more difficult to change: stability with time and address space skewness [32–34]. If it can be proven that the IP address used by a domain name *d* is positively related to a known malicious activity, then the domain name may be considered as malicious. Considering these two characteristics, the *Segugio* [14] and *Doctrina* [15] approaches successfully transformed the correspondence between the IP and domain names into features to mine for malicious domain names from the DNS logs. Moreover, some research used domain IP mapping as a trait to find

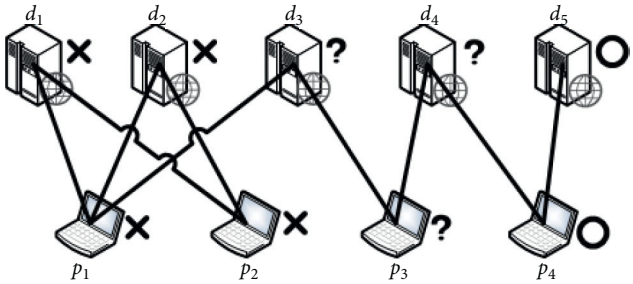

Figure 1: *Process*-domain annotated graph.

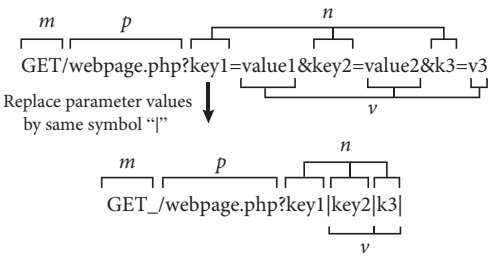

Figure 2: Common indication to simplify the HTTP request to *Trace*.

network threats [35, 36]. While these detection methods can still be improved, they prove that IP addresses could be an effective identification factor. The *MD-Miner*^{P} takes advantage of the mapping between the domain and IP address to become the third factor. This approach employs the interaction profiling bipartite graph concept to construct the IP-domain interaction profiling graph from network traffic logs to produce effective detection features.

2.4. Lexical Analysis. Manipulating the domain name is another common practice for cybercrimes. Previous research [37] has shown that nearly one-third of all websites in the world are potentially malicious. Many malicious URLs follow obfuscation methods that make the URL strings similar to benign URLs to avoid detection. However, studying various detection methods by analyzing the diversity of domain strings allows designing effective malicious URL detection solutions [38]. These develop lexical features that excavate the divergence of URLs by analyzing the statistical properties of URL strings. The adjective lexical describes the relation to a vocabulary of words and the associated lexical analysis is based on the characteristics of the URL string to determine the lexical features that represent the features of a URL name. Lexical features refer to the actual text without other external information of the URL string. The intention is to make malicious URLs "look" different to experts when compared with benign ones [27].

Most lexical features commonly used for such classifications include the statistical properties of the URL string, like the numerical information regarding the feature lengths (URL length, top-level domain length, primary domain length, etc.) and the number of special characters [39]. The extracted information is obfuscation-resistant and useful. One lexical analysis approach is called the bag-of-words (BoW), which builds a dictionary as a feature set by referring to all the different types of words in all URLs. When a URL includes a word in the dictionary, the value of the feature is 1; otherwise, it is 0. The *MD-Miner*^{P} developed a kind of BoW approach to adapt to big data and accelerate the computing, which is described in detail in Section 3.

Due to the lack of scalability of previous research [26–30], this was restricted to a small amount of material. Therefore, this paper proposes a *MD-Miner*^{P} system which is mainly used to extract hidden malicious threats from long period and large amount of network traffic logs. Our approach takes full advantage of the concept that Internet communications for a specific purpose will invoke similar interactions. The *MD-Miner*^{P} uses attributes in the network traffic log to create representative characteristics for each domain, which answers four important questions. (1) Who is connecting what (*Process*-domain interaction profiling graph)? (2) How to interact with what (*Trace*-domain interaction profiling graph)? (3) What domain name used what IP (IP-domain interaction profiling graph)? (4) What does it "look" like (lexical analysis)? An exhaustive description of how to use the unique methodologies proposed in this paper to establish effective classification features for each domain is given in the following section.

3. *MD-Miner*^{P} Implementation

The concept of the *MD-Miner*^{P} is to track known and discover unknown malicious network domains, which are designated as a channel for attackers to perform malicious acts. Looking at network communications from this perspective allows finding similar traces of connections, and the victim machine generally attempts to connect to malicious or newly created domains. Therefore, the *MD-Miner*^{P} is based on the following main intuitions:

(1) Victim clients tend to connect malicious domain families.

(2) Malware belonging to the same family tend to connect to partially overlapping malware-controlled domains.

(3) Benign applications rarely connect to domains that exist only to provide malicious functionality.

(4) Cybercriminals prepare multiple malicious domains to prevent single-point failure.

(5) Malicious domains reuse the same IP addresses.

(6) Domain names with the same purpose often "look the same."

To take advantage of these points, we proposed a new malicious domain detection system called *MD-Miner*^{P}. The first part of this section gives a detailed explanation for the capture of network domain features. The second part elaborates on the implementation details of the *MD-Miner*^{P} based on the MapReduce framework.

3.1. Domain Features. For each domain in the network traffic, the *MD-Miner*^{P} creates four feature vectors. Three

feature vectors are generated based on the interaction profiling bipartite graph, and the other feature vector is generated based on the lexical analysis. The intuitions described in Section 2 are used to generate relevant features through the interaction profiling bipartite graph, as shown in Figure 3.

In the interaction profiling bipartite graph, the domain represents the node on one side of the binary graph, and the *CF* stands for "connection factor," which is the node on the other side. The connection factors include the *Process*, *Trace*, and *Address* used by the domain. The *Process* indicates the user-agent plus client-IP, *Trace* indicates the simplified HTTP request, and *Address* indicates the domain IP address. The *MD-Miner*^{P} defines three interaction profiling bipartite graphs using these connection factors, where $G_P = (P, D, E_{PD})$ represents the interaction profiling bipartite graph for *Process*, $G_T = (T, D, E_{TD})$ is for *Trace*, and $G_A = (A, D, E_{AD})$ is for *Address*. Node set D represents the domain nodes with $d_i \in D$, node set P represents the *Process* nodes with $p_i \in P$, node set T represents the *Trace* nodes with $t_i \in T$, and node set A represents the *Address* nodes with $a_i \in A$. The edge sets are called E_{PD} in G_P, E_{TD} in G_T, and E_{AD} in G_A. The *Process* p_i connects a domain d_j with an edge $e_{ij} \in E_{PD}$, the *Trace* t_i connects a domain d_j with an edge $e_{ij} \in E_{TD}$, and the *Address* a_i connects a domain d_j with an edge $e_{ij} \in E_{AD}$. The features of different aspects of the network domain can be described from the interaction profiling bipartite graphs for different *CF*s. Communications with the same purposes interact through similar *CF*s. For example, the d_1 and d_2 conduct similar communications as shown in Figure 3. Once the interaction profiling bipartite graph is constructed, the next step is to extract the domain feature vectors from each graph, as detailed below.

Each domain name needs to go through three phases to extract the feature vector by analyzing the interaction profiling bipartite graph. The first phase is to mark the domain node, which obtains benign and malicious domain intelligence (whitelist/blacklist) from a public or private reputation database. If the domain exists in the whitelist, it is marked as *DomainWhite*; if it exists in the blacklist, it is a known malicious domain and marked as *DomainBlack*. All remaining domains are marked as *DomainUnknown*, which are the primary targets for further classification to mine malicious domains that are not recorded in the threat intelligence but are actually hidden.

The second phase is to label each *CF* node as *White*, *Black*, *Mix*, *Unknown*, or *Leaf*. The labeling method is based on the labeled domain nodes where each *CF* node is linked. Three numbers are counted for each *CF* node, namely, $White_{sum}$, $Black_{sum}$, and $Unknown_{sum}$, These are the number of edges of a *CF* node connected to different *DomainWhite*, the number of edges for different *DomainBlack*, and the number of edges for different *DomainUnknown*. Each *CF* node in the interaction profiling bipartite graph is then labeled with its own $White_{sum}$, $Black_{sum}$, and $Unknown_{sum}$. The labeling method is as follows, where the *CF* nodes in the lower part of Figure 4 illustrate the labeling method.

(1) *White*: $White_{sum} > 0$ & $Black_{sum} = 0$ (circle)

(2) *Black*: $\text{Black}_{\text{sum}} > 0$ & $\text{Black}_{\text{sum}} > \text{White}_{\text{sum}}$ (cross)

(3) *Mix*: $Blacksum > 0$ & $White_{sum} > Black_{sum}$ (circle sign combined with cross)

(4) *Unknown*: $Black_{sum} = 0$ & $White_{sum} = 0$ (question mark)

(5) *Leaf*: Connect to a single domain only (triangle)

The third phase is to compute the feature values for each domain node. Figure 4 shows the interaction profiling bipartite graph G_P, where the *Process* feature values of the domain node d_3 are calculated using the G_P as an example. Five values are counted from the attributes of the labeled *Process* nodes to which d_3 is linked: S_P, W_P, B_P, M_P, and U_P, where S_P is the total number of *Process* nodes linked to d_3; W_P is the number of *Process* nodes linked to d_3 and labeled as *White*; B_P is the number of *Process* nodes linked to d_3 and labeled as *Black*; M_P is the number of *Process* nodes linked to d_3 and labeled as *Mix*; and U_P is the number of *Process* nodes linked to d_3 and labeled as *Unknown*. The six following *Process* feature values of d_3 are calculated using the following formulas.

(1) Fraction of *White Process* nodes, $w_P = |W_P|/|S_P|$

(2) Fraction of *Black Process* nodes, $b_P = |B_P|/|S_P|$

(3) Fraction of *Mix Process* nodes, $m_P = |M_P|/|S_P|$

(4) Fraction of *Unknown Process* nodes, $u_P = |U_P|/|S_P|$

(5) Fraction of *Leaf Process* nodes, $l_P = |L_P|/|S_P|$

(6) Fraction of *total Process* nodes, $s_P = |S_P|$

The feature values for d_3 obtained from the above six formulas are 1⁄6, 2⁄6, 1⁄6, 1⁄6, 1⁄6, and 6. Following the same pattern applied to the interaction profiling bipartite graph G_T allows using d_3 to obtain w_T, b_T, m_T, u_T, l_T, and s_T. Applying this to the interaction profiling bipartite graph G_A allows using d_3 to obtain w_A, b_A, m_A, u_A, l_A, and s_A. All the domain nodes are assigned their own 18 feature values in the same way.

The lexical features are those acquired based on the properties of a domain name or string. The motivation is that the domain-based "appearance" should be able to identify the malicious nature of a domain. The *MD-Miner*^{P} directly uses the BoW model, which loses information on the order of tokens that belong to the top-level and primary domains. This is done by creating a separate dictionary for each fragment. The lexical features also include the statistical properties of the domain, such as the length of its name and the number of "." characters.

3.2. MapReduce Algorithm. The *MD-Miner*^{P} is based on two important phases to detect potentially malicious domains, as shown in Figure 5: domain feature extraction and random forest classifier. First, the *MD-Miner*^{P} constructs the domain node feature vector by taking the network traffic log and benign\malicious domain intelligence stored in the domain threat intelligence database as inputs. The domain feature extraction phase consists of four parts that extract 22 features of each domain node: (1) *Process*, (2) *Trace*, (3) *Address*, and (4) lexical feature extractions. Second, the *MD-Miner*^{P} adopts Spark parallel processing to build a random forest

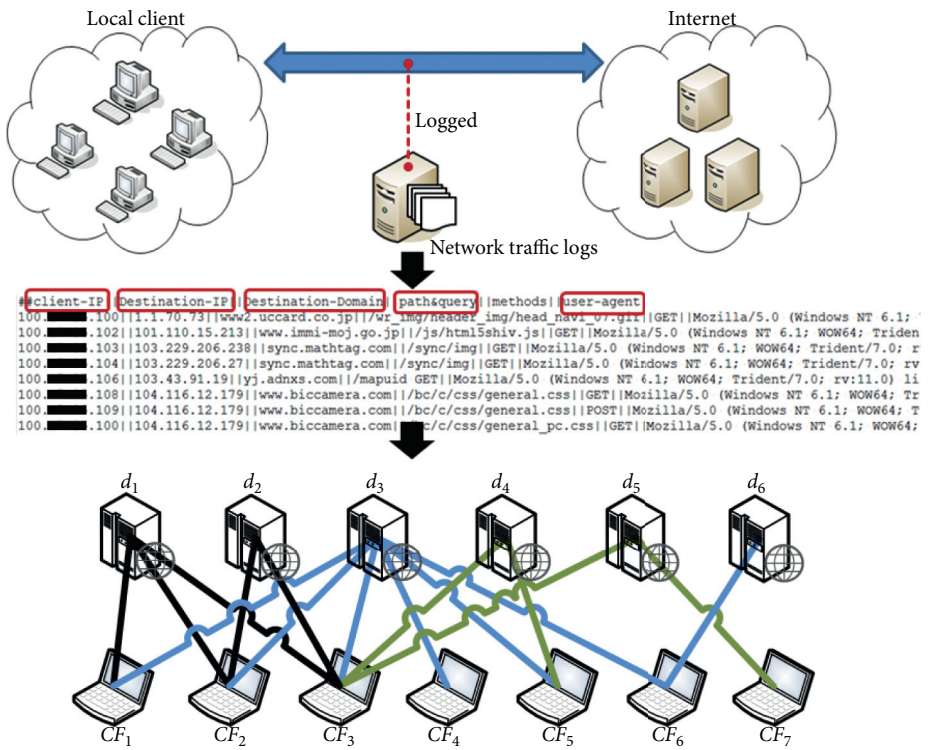

Figure 3: An illustration of the conversion of network traffic into the interaction profiling bipartite graph.

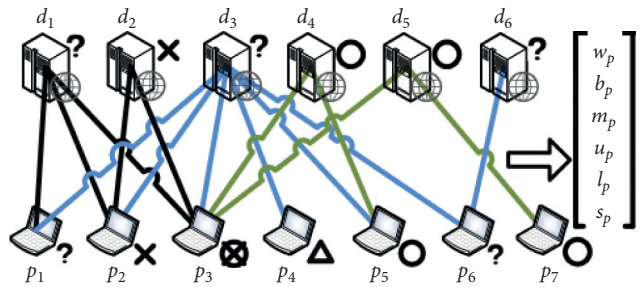

Figure 4: Illustration of the process feature vector generated from the interaction profiling bipartite graph.

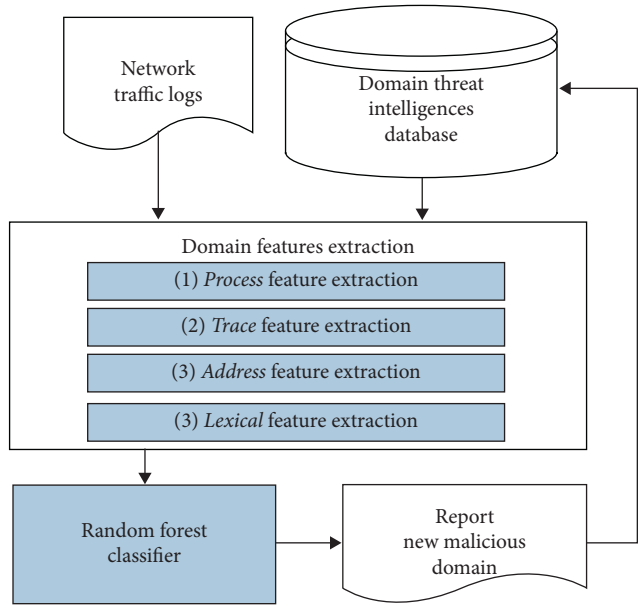

Figure 5: Flow diagram for the *MD-Miner*^{P} process.

classifier based on the decision tree model, which is employed to detect malicious domains.

Parts (1)–(3) of the domain feature extraction are based on a similar concept of using the interaction profiling bipartite graph to obtain adjacent information as features. The *MD-Miner*^{P} designs four MapReduce jobs to realize feature extraction of the interaction profiling bipartite graph: (1) domain node labeling, (2) *CF* node labeling, (3) interaction profiling bipartite graph building, and (4) behavior feature calculating. Taking part (1) as an example, the following is a detailed description of the MapReduce jobs for the *Process* feature extraction when the *Process* nodes are used as *CF* nodes.

The domain node labeling job first utilizes multiple input mechanisms of the map phases with the network traffic and whitelist/blacklist as the input and domain as the key. Parallel label domain nodes are either *DB* (*DomainBlack*), *DW* (*DomainWhite*), or *DU* (*DomainUnknown*) in the reduce phase based on shuffle and sorting mechanisms. An example of the data flow for domain node labeling is shown in Figure 6.

The next job after labeling the domain nodes is to label the *CF* nodes. As described in Section 3.1, the label of a *CF* node is determined from the connected domain nodes. The five label types are *White*, *Black*, *Mix*, *Unknown*, and *Leaf*. The input to the *CF* node labeling job is the output of the domain node label from the previous step. Therefore, the MapReduce job at this step takes the *CF* (e.g., *Process*) node as the key and the domain node as the value in the map phase. In the reduce phase, the number of occurrences for *DB*, *DW*, and *DU* for each *CF* node are counted and the corresponding labels are calculated. The *Process* nodes are taken as the *CF* nodes as an example, and Figure 7 shows the data flow of the labeled *CF* nodes.

The next job is to build the interaction profiling bipartite graph to aggregate the labeled domain nodes and labeled *CF* nodes into a dataset. In the map phase, the output of the domain and *CF* node labeling jobs are taken as the inputs to use the advantages of multiple input mechanisms with the identity of the *CF* (e.g., *Process*) node as the key. The *CF* node labels are annotated for each record to obtain the interaction profiling bipartite graph in the reduce phase. Figure 8 shows an example of the data flow to build an interaction profiling bipartite graph during this job.

The interaction profiling bipartite graph constructed in the above jobs allows calculating the behavior features for each domain node. In the map phase, the constructed interaction profiling bipartite graph output from the previous job is taken as the input, where the domain node is the key. In the reduce phase, each domain node obtains its neighbor's information (labels of *CF* nodes) through the shuffle and sorting mechanism. Therefore, the *MD-Miner*^{P} can compute the behavior features of each domain node in parallel. Figure 9 shows an example of the parallel computing behavior features in the job.

Parts (2) and (3) can be implemented as similar MapReduce jobs for G_T and G_A. The only difference is that Part (1) uses the *Process* (user-agent + client-IP) and domain nodes to construct the interaction profiling bipartite graph G_P, Part (2) uses the *Trace* nodes instead of the *Process* nodes to build the interaction profiling bipartite graph G_T, and Part (3) uses the *Address* (destination IP address) nodes to replace the *Process* nodes and construct the interaction profiling bipartite graph G_A.

The lexical feature extraction in Part (4) uses distributed caching mechanisms to store dictionaries for both the primary and top-level domains and gives each term an index number. The distributed caching mechanism allows calculating the lexical features in a single map phase, including the length of the domain, the number of "." characters, and the index numbers of the top-level and main domains.

Once each domain in the dataset has its own 22 feature values based on the above steps, the *MD-Miner*^{P} performs two steps to employ the random forest classifier based on Spark, which is a unified analytics engine for large-scale data processing. The first step constructs a classifier RF_C by taking all the *DB*, *DW*, and their feature values in the dataset as the training set and inputs them into the random forest algorithm. The second step is to use the classifier RF_C to identify all unknown domains labeled as *DU* in the dataset.

4. Evaluation

The *MD-Miner*^{P} mines stealthy malicious domains for enterprise-scale big network traffic data. Therefore, the *MD-Miner*^{P} is deployed for enterprise network environments. The deployed network environments are called ENT_{N1} and ENT_{N2}, which are both real-world companies based in Taiwan with thousands of networked clients that install and run antivirus software. The ENT_{N1} is a medium-scale company and its compliance with security management rules is relatively relaxed. The organization's network traffic was collected for 8 months (Jan 1, 2018, to Aug 31, 2018). The ENT_{N2} is a large-scale company that follows strict security and information management regulations with a collected network traffic period of 2 weeks (Aug 1, 2018, to Aug 15, 2018). Table 1 gives further details for both datasets.

The experiment presented in this paper is based on the two large network traffic datasets ENT_{N1} and ENT_{N2} and evaluates the overall performance of the *MD-Miner*^{P} from three perspectives. First, k-fold cross-validation was employed to evaluate the classification capabilities of *MD-Miner*^{P}. Second, the actual instances demonstrate the ability of *MD-Miner*^{P} to mine hidden malicious domains. Finally, the ability of *MD-Miner*^{P} to handle big data is demonstrated by adjusting the number of nodes in the parallel computing cluster and observing its operational performance.

To perform the k-fold cross-validation, we begin by marking all the known samples in the dataset as n (negative) or p (positive), where n is interpreted as benign and p is interpreted as malicious. A prediction result produced from classifying a sample with the model is divided into four types. First, true positive (*TP*) indicates the result of the classifier to predict the sample is p when it is; second, false positive (*FP*) indicates the result of the classifier predicts the sample is p when it is n; third, true negative (*TN*) indicates the result when the classifier predicts the sample is n when it is; and fourth, false negative (*FN*) indicates the result when the classifier predicts the sample is n when it is p.

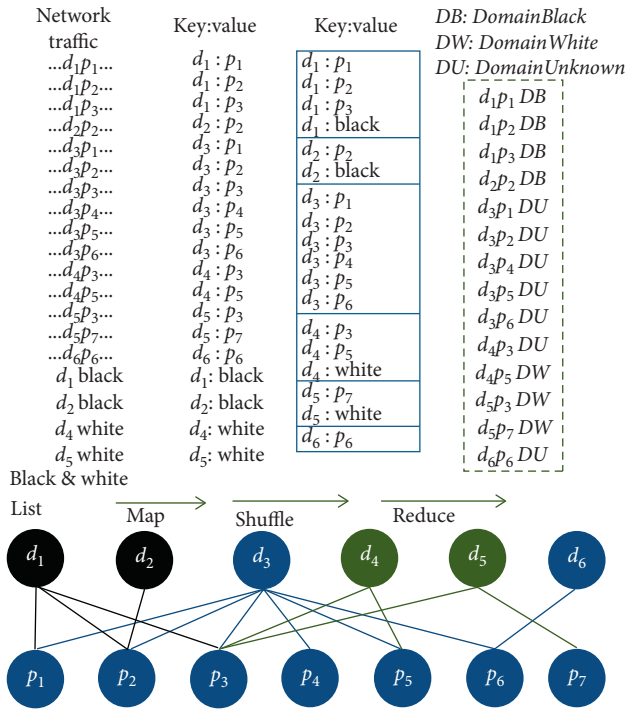

Figure 6: Illustration of domain node labeling for MapReduce jobs.

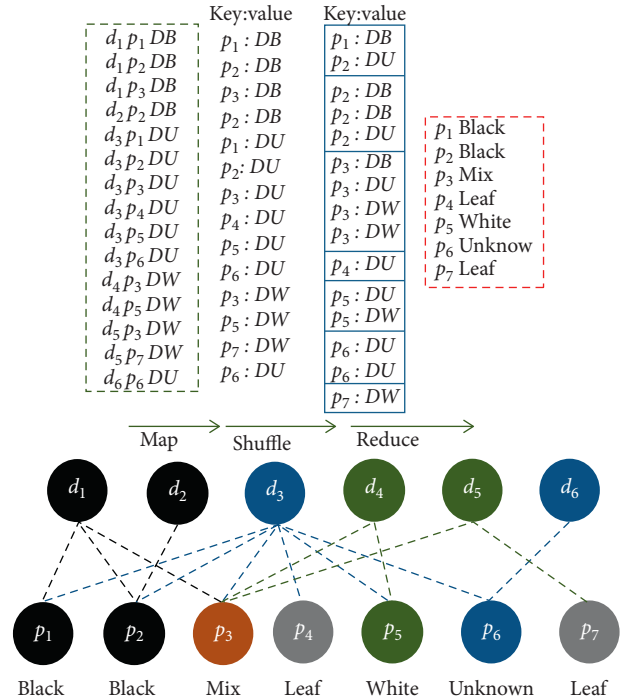

Figure 7: Illustration of the *Process* node labeling for the MapReduce job.

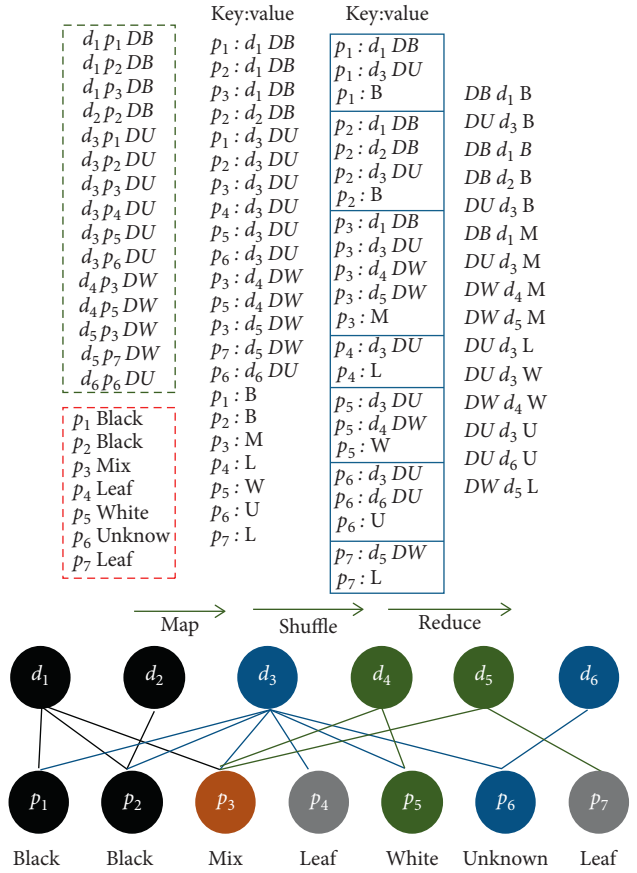

Figure 8: Illustration of interaction profiling bipartite graph building for the MapReduce job.

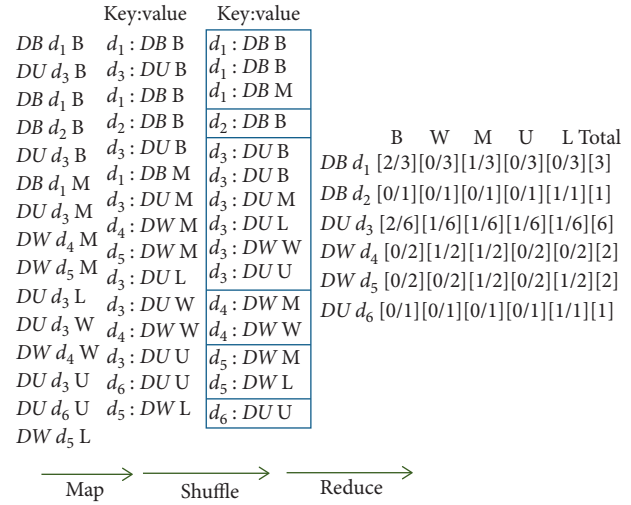

Figure 9: Illustration of the process behavior feature computation for the MapReduce job.

The above description indicates that the first step to prepare the benchmark is to label the ENT_{N1} and ENT_{N2} datasets. Labeling *DW* required employing commercial and public intelligence as whitelists, such as Bluecoat, and the collection of the top 1 million most famous domain names from alexa.com. Labeling the *DB* data required checking if the entire domain name string matches an existing domain from a commercial domain blacklist. When the tagged datasets (ENT_{N1} and ENT_{N2}) are ready, we can generate evaluation criteria for different feature vectors. To conduct a

Table 1: Attributes of the experimental datasets.

	Size (GB)	No of Client-IPs	No of destination domains	No of HTTP requests
ENT_{N1}	34.2	2,336	4,912,810	230,350,760
ENT_{N2}	172.7	13,829	11,946,898	442,036,055

comprehensive evaluation, different metrics are needed: precision, recall, F-measure, accuracy, and AUC. Furthermore, each metric is calculated through a cross-validation process.

The k-fold cross-validation is a resampling procedure used to evaluate machine learning models for limited data samples. The original samples are randomly divided into k equally sized subsamples, where a single subsample is retained as the data for the validation model, while the other k-1 subsamples are used to train the model. The cross-validation process is then repeated k times (called folds), with each k subsample being used only once as the verification data. The results of the k-folds can then be averaged to produce a single estimate. The advantage of the k-fold cross-validation process is that each data sample only needs to be tested once and used to train k-1 times [40]. This paper adopts the 10-fold cross-validation procedure.

4.1. 10-Fold Cross-Validation for MD-Miner^P. The first experiment deployed *MD-Miner*^{P} to the real-world ENT_{N1} and ENT_{N2} datasets and determined their 10-fold cross-validation results. The *MD-Miner*^{P} constructed three interaction profiling bipartite graphs (G_P, G_T, and G_A) by applying the feature extraction method described in Section 3. The three feature vectors were then generated using G_P, G_T, and G_A; each feature vector contained six feature values. Moreover, we used the proposed lexical analysis method to generate the fourth feature vector that contains four feature values. In addition, a feature vector containing 22 feature values was generated by merging the above four feature vectors. The performance of each feature vector was confirmed by observing the classification results calculated by different metrics as shown in Table 2. In addition, we used the ROC curve to show the ability of the classification model to all classification thresholds as shown in 10 and 11

The above experimental results show that when the *MD-Miner*^{P} was deployed to the ENT_{N1} dataset, the *Address* feature vector performed the best, which the AUC and F-measure were as high as 0.99. Although the AUC of the other three feature vectors is greater than 0.8, the recall metric is low, indicating that these features are only applicable to partial data. However, combining feature vectors can improve the overall ability of classification. When *MD-Miner*^{P} was deployed in ENT_{N2} dataset, the characteristic of combining features resulted in a more significant increase in overall classification capacity. Since ENT_{N2} belongs to a relatively diversified dataset, the recall value of general feature vectors is low. However, by combining the feature vectors, the recall can be significantly improved. Furthermore, in both the ENT_{N1} or ENT_{N2} datasets, the AUCs of the feature vectors that combined the other four were above 0.98, which indicates outstanding discrimination.

4.2. Interaction Profiling Bipartite Graph versus Annotated Bipartite Graph. The second experiment was to prove that the interaction profiling bipartite graph leveraged here is better than the annotated bipartite graph adopted in previous studies [12, 14, 15], which are compared in Figure 12. The annotated bipartite graph only considers the benign and malicious attributes of the connected domain when extracting features, as described in Section 3. The interaction profiling bipartite graph further considers additional aspects, such as outlier domains. Therefore, for the same *CF*, the annotated bipartite graph exports three feature values, while the interaction profiling bipartite graph brings out six feature values.

The experiment also utilized the ENT_{N1} and ENT_{N2} datasets. The annotated and interaction profiling bipartite graphs were formed using the same datasets and the same *CF* (selected from *Process*, *Trace*, and *Address*) to generate three feature vectors, which were combined into a fourth feature vector. Comparing the 10-fold cross-validation AUCs of the four feature vectors generated from the annotated bipartite graph and interaction profiling bipartite graph shows which bipartite graph had a better recognition effect. It is noted that the lexical feature vector is not included in the experiment because it only compares two bipartite graphs. The implementation of the annotated bipartite graph is based on previous studies [12, 14, 15], and the interaction profiling bipartite graph is defined in Section 3.

Figures 13–16 are the results of experiments based on the ENT_{N1} dataset. Figure 13 shows that, for the *Process CF*, the annotated bipartite graph had an AUC of 0.85 and the interaction profiling bipartite graph had an AUC of 0.85. Figure 14 shows that, for the *Trace CF*, the annotated bipartite graph had an AUC of 0.74 and the interaction profiling bipartite graph had an AUC of 0.80. Figure 15 shows that, for the *Address CF*, the annotated bipartite graph had an AUC of 0.62 and the interaction profiling bipartite graph had an AUC of 1.00. Figure 16 shows the experiment that combined the feature values generated by the *Process*, *Trace*, and *Address CF*s gave AUCs for the annotated bipartite graph and interaction profiling bipartite graph of 0.94 and 1.00, respectively.

Figures 17–20 are the results of experiments based on the ENT_{N2} dataset. Figure 17 shows that, for the *Process CF*, the annotated bipartite graph had an AUC of 0.79 and the interaction profiling bipartite graph had an AUC of 0.97. Figure 18 shows that, for the *Trace CF*, the annotated bipartite graph had an AUC of 0.54 and the interaction profiling bipartite graph had an AUC of 0.75. Figure 19 shows that, for the *Address CF*, the annotated bipartite graph had an AUC of 0.68 and the interaction profiling bipartite graph had an AUC of 0.96. Figure 20 shows the experiment that combined the feature values generated from the *Process*, *Trace*, and *Address CF*s. The annotated and interaction

Table 2: Cross-validation results of MD-Miner^P for different feature vectors of ENT_{N1} and ENT_{N2}.

Dataset	Feature vectors	Precision (%)	Recall (%)	F-measure (%)	Accuracy (%)	AUC (%)
ENT_{N1}	Process feature	97.42	28.41	43.96	95.42	83.93
	Address feature	100.00	99.72	99.86	99.98	99.97
	Trace feature	97.35	21.27	34.88	94.98	79.61
	Lexical feature	64.57	20.20	30.54	94.23	85.89
	Merged feature	100.00	99.85	99.92	99.99	99.97
ENT_{N2}	Process feature	63.19	10.85	17.74	99.82	96.88
	Address feature	100.00	48.05	63.60	99.90	95.86
	Trace feature	84.44	28.90	42.70	99.86	75.29
	Lexical feature	85.83	12.67	21.45	99.83	71.29
	Merged feature	94.04	64.38	75.28	99.93	97.34

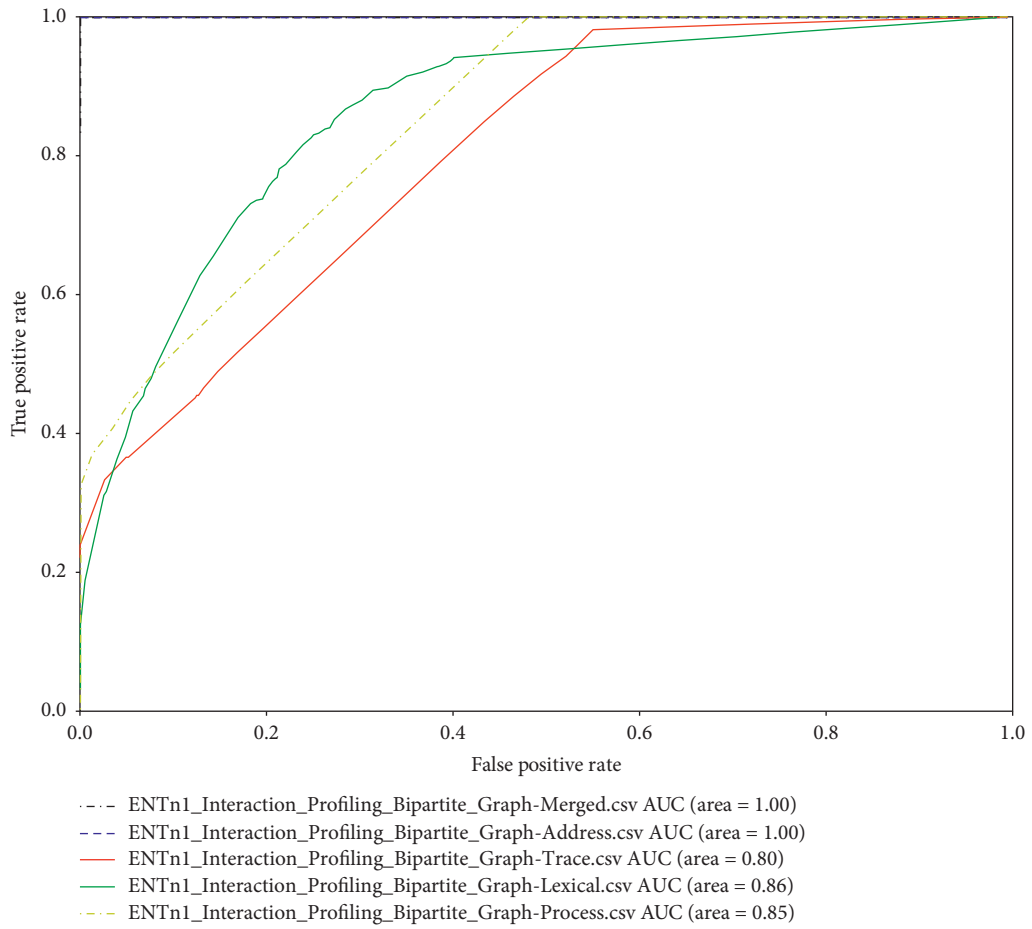

Figure 10: Illustration of the ROC curves for the five feature vectors generated from the ENT_{N1} dataset.

profiling bipartite graphs had AUCs of 0.83 and 0.95, respectively. Table 3 summarizes the AUC value of the feature vector generated by the annotated bipartite graph and interaction profiling bipartite graph in respect of classification assessment, which is used to evaluate the ability of classification.

The experiments in this section show that the data for the proposed interaction profiling bipartite graph are superior to the annotated bipartite graph in either deployed network environments of ENT_{N1} or ENT_{N2}.

4.3. Identified Malicious Domain Analysis. This section demonstrates the effectiveness of the *MD-Miner*^{P} at mining potentially malicious domains. With unknown domains in the ENT_{N2} dataset as the objects for detection, Table 4 shows the top 10 domains with the highest malicious probability as detected with the *MD-Miner*^{P}. These 10 domains were analyzed using the VirusTotal, and four were identified as malicious while the remaining six were classified as clean. Due to the limited space for digital forensics content of the domain name "folder[.]maroon91[.]com," this section

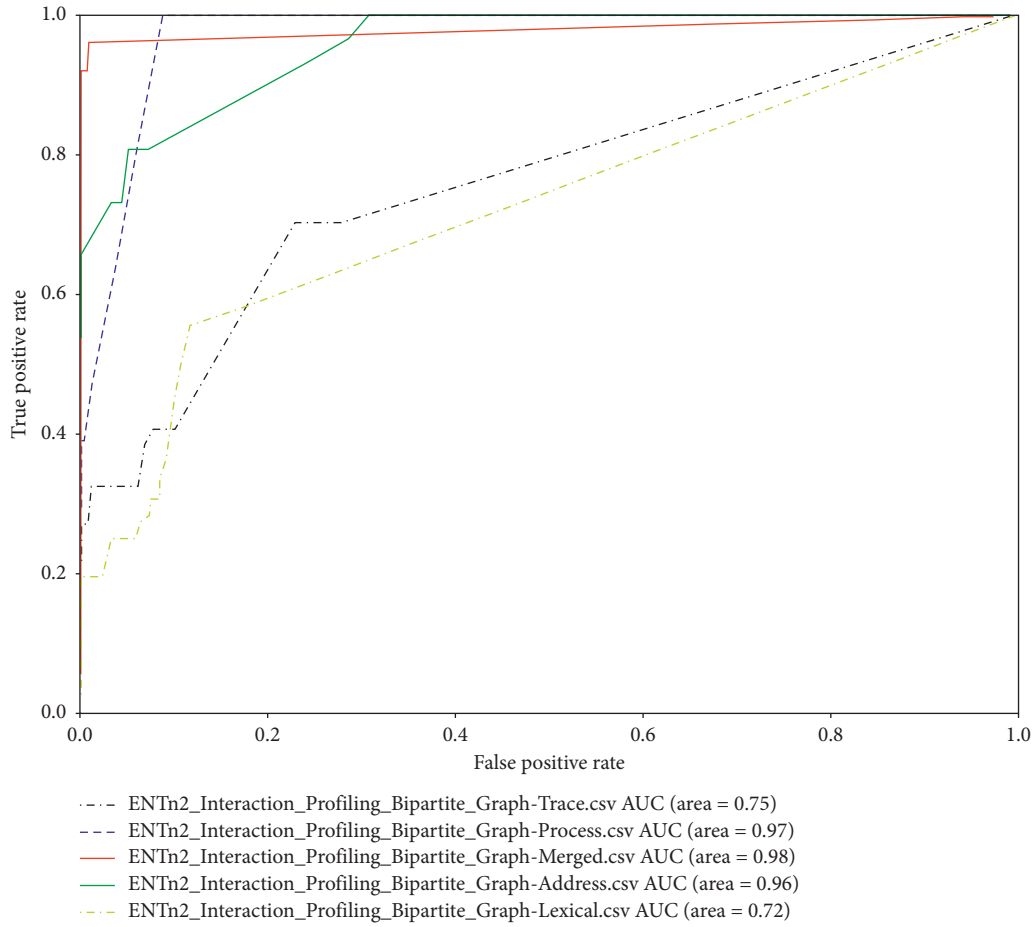

Figure 11: Illustration of the ROC curves for the five feature vectors generated from the ENT_{N2} dataset.

describes the digital forensics as an example. The evidence to classify the domain as malicious is collected from external threat intelligence, including the file communication records and passive DNS records, and the relationship graph regarding the domain based on the collected information is created.

The relationship graph of the domain "folder[.]maroon91[.]com" is constructed using CyberGraph [13], as shown in Figure 21. The figure shows the direct and indirect relationship between the domain and malicious files, malicious IP, and malicious domains. The domain "folder[.]maroon91[.]com" is not a malicious site in VirusTotal, but the malicious files are directly connected to the domain. The domain is then resolved as "221[.]228[.]214[.]69," "58[.]215[.]186[.]83," "118[.]193[.]145[.]130," and "118[.]193[.]187[.]35." The records of these IP addresses that communicate with malicious files are observed from the graph. Moreover, the domains "dsc[.]maroon91[.]com," "app[.]maroon91[.]com," "usjzx[.]maroon91[.]com," "upgrade[.]maroon91[.]com," and "folderhw[.]maroon91[.]com" were discovered to communicate with the malicious files, which have a domain sibling relationship with each other. A series of outward relationships helped identify the domain "folder[.]maroon91[.]com" as malicious. As a consequence, this section demonstrates that the *MD-Miner*^{P} can detect malicious domains that are not recognized by other reputable intelligence systems.

4.4. Performance Evaluation. From a complexity theory viewpoint, the MapReduce framework is unique in that it combines bounds on time, space, and communication. Each of these bounds would be very weak on its own: the total time available to processors is polynomial; the total space and communication are slightly less than quadratic. In particular, even though arranging the communication between processors is one of the most difficult parts of designing a MapReduce algorithm, classical results from communication complexity do not apply since the total communication available is more than linear [41]. Therefore, we use fixed dataset to measure the execution performance and scalability of MapReduce through the execution time of different cluster sizes.

The performance and scalability of the *MD-Miner*^{P} are verified by adjusting the number of nodes in the Hadoop cluster, which were two, four, and six. Each node had 24 CPUs (each is an Intel (R) Xeon (R) CPU E5-2620 2.00 GHz processor) with 32 GB of RAM. The dataset used as a benchmark to analyze the *MD-Miner*^{P} runtime is the ENT_{N2} dataset described in Table 1, which is sized at 172.7 GB.

The flow of the *MD-Miner*^{P} can be divided into three parts: data preprocessing, feature extraction, and domain classification. The feature extraction stage of the *MD-Miner*^{P} can be classified into two parts: interaction profiling bipartite

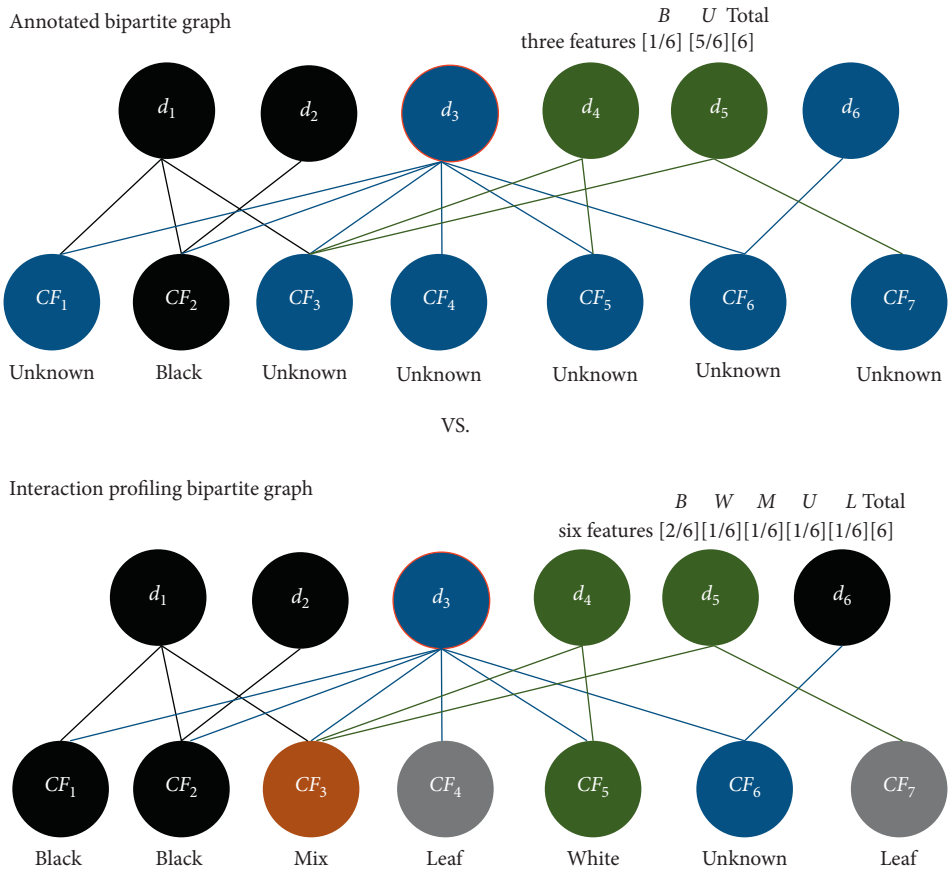

Figure 12: Difference between feature extractions for the annotated and interaction profiling bipartite graphs.

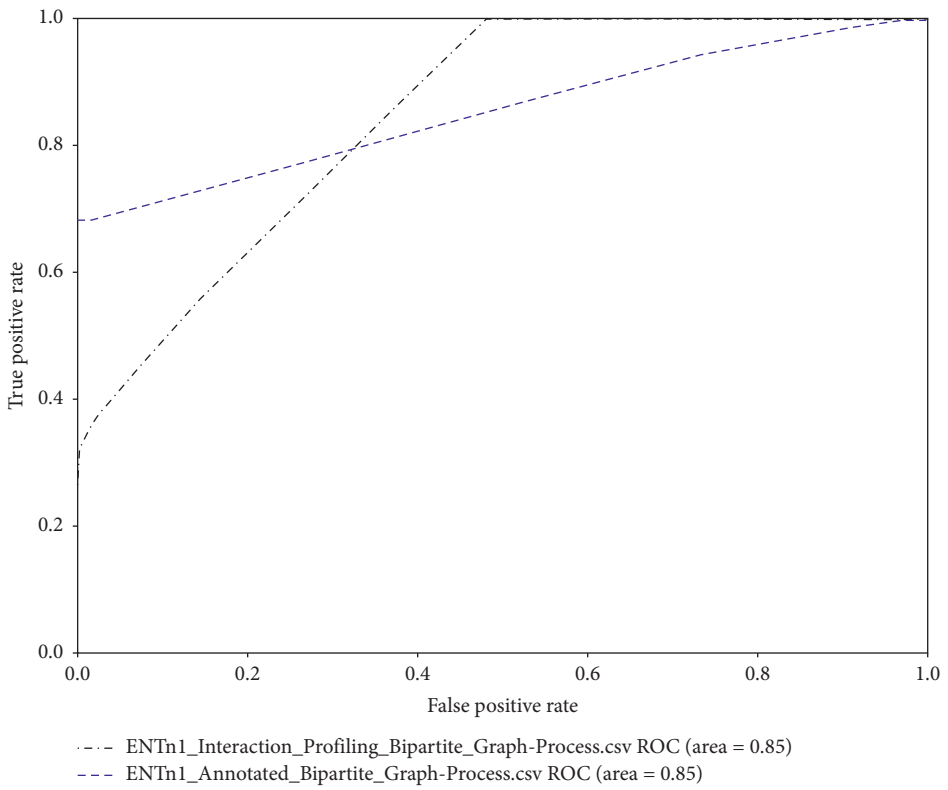

Figure 13: Illustration of the ROC curve for the two *Process* feature vectors generated from two kinds of bipartite graphs for the ENT_{N1} dataset.

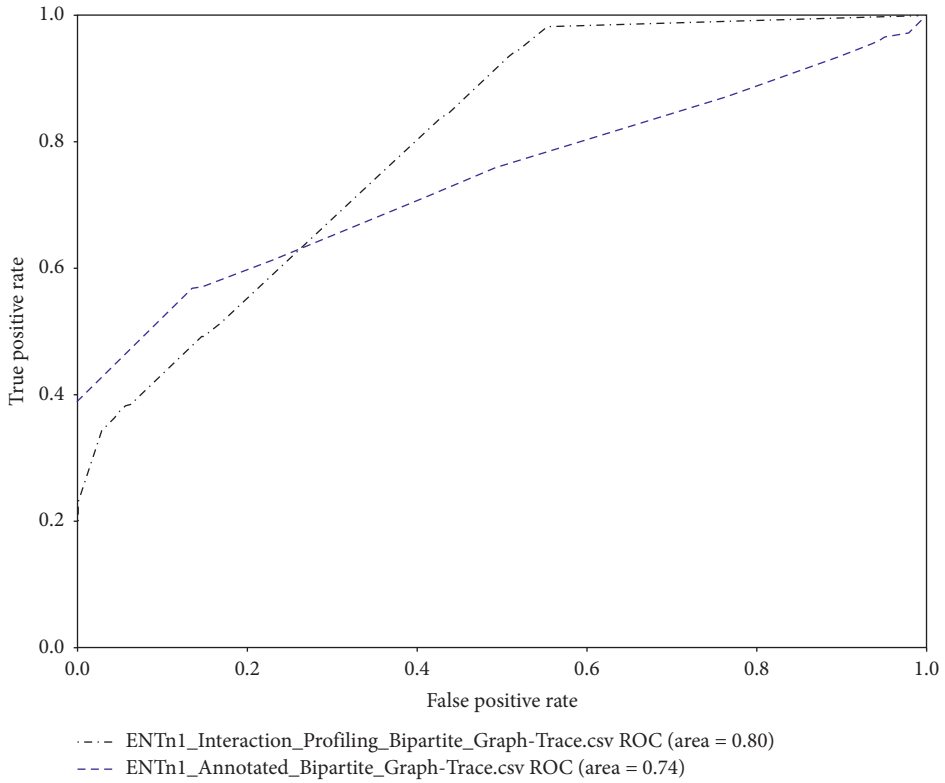

Figure 14: Illustration of the ROC curve for the two *Trace* feature vectors generated from two kinds of bipartite graphs for the ENT_{N1} dataset.

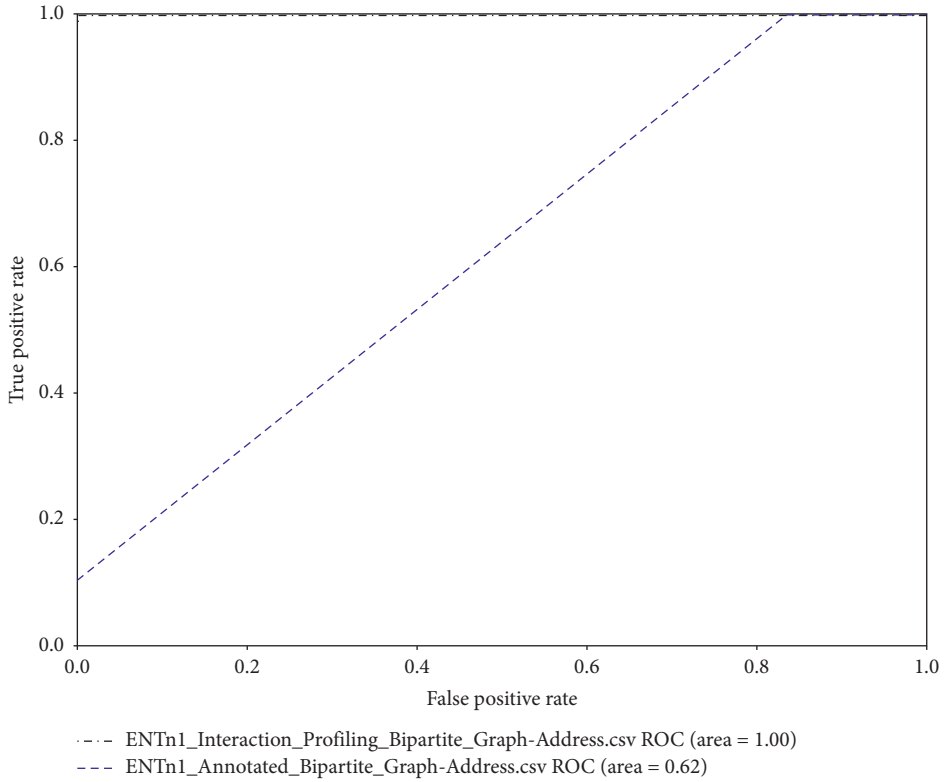

Figure 15: Illustration of the ROC curve for the two *Address* feature vectors generated from two kinds of bipartite graphs for the ENT_{N1} dataset.

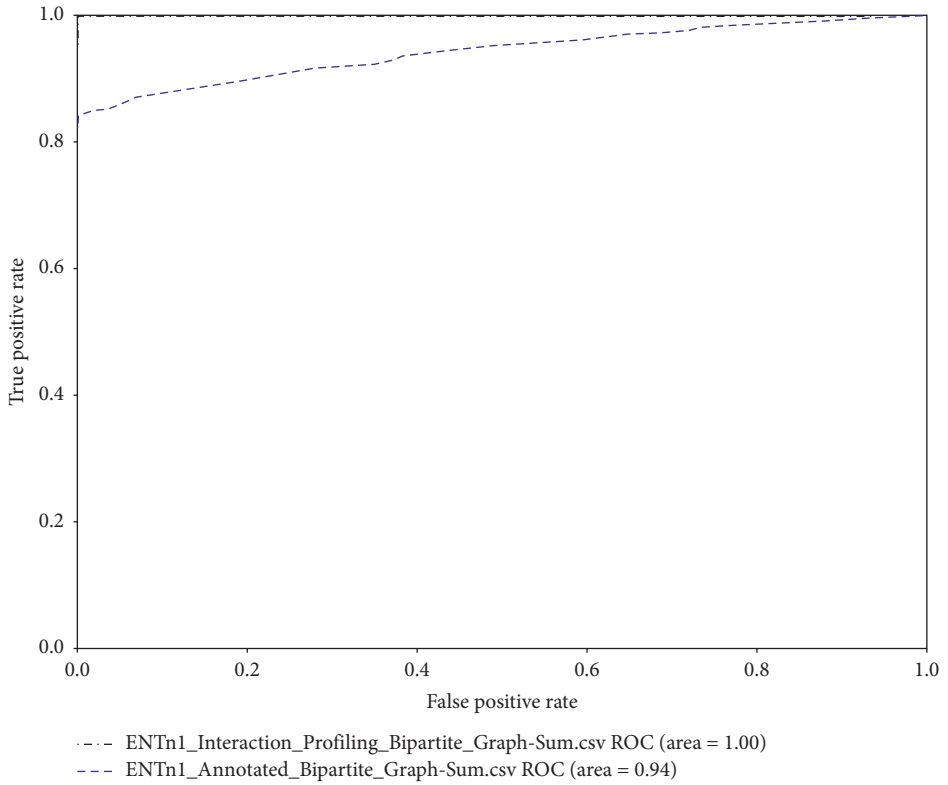

Figure 16: Illustration of the ROC curve for the combined three feature vectors generated from the two bipartite graphs for the ENT_{N1} dataset.

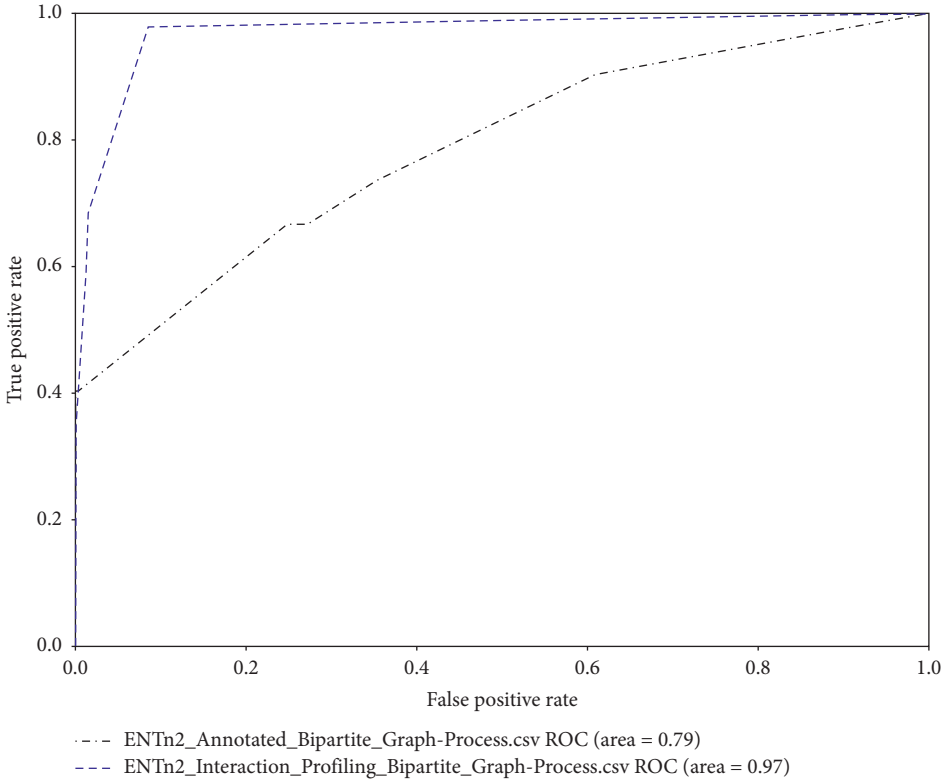

Figure 17: Illustration of the ROC curve for the two *Process* feature vectors generated from two kinds of bipartite graphs for the ENT_{N2} dataset.

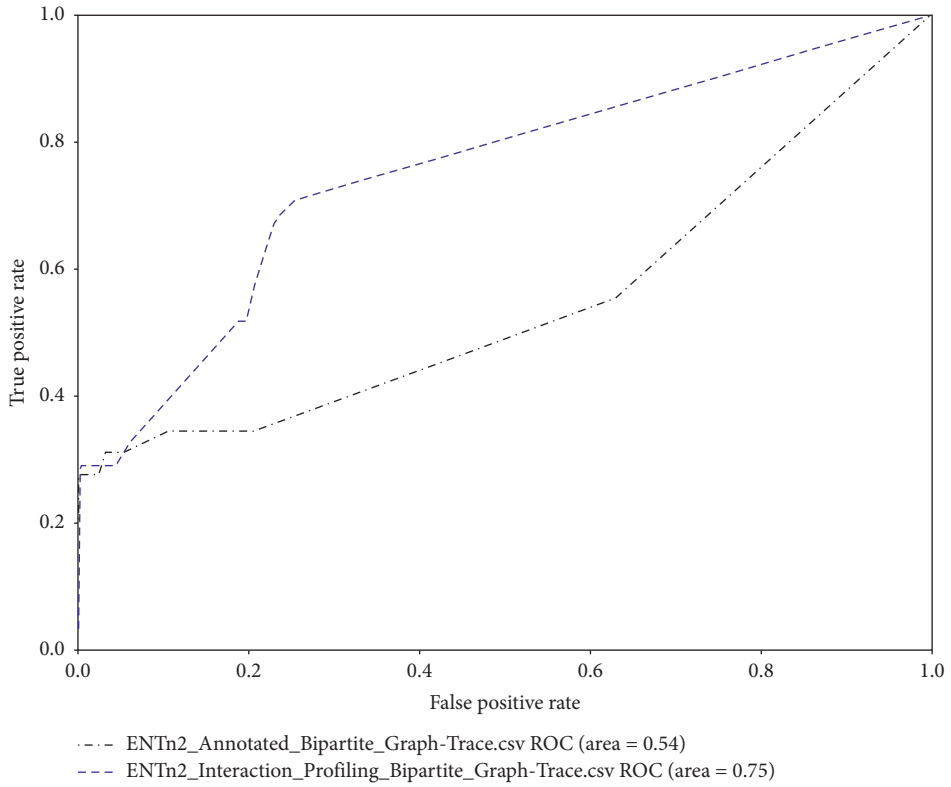

Figure 18: Illustration of the ROC curve for the two *Trace* feature vectors generated from two kinds of bipartite graphs for the ENT_{N2} dataset.

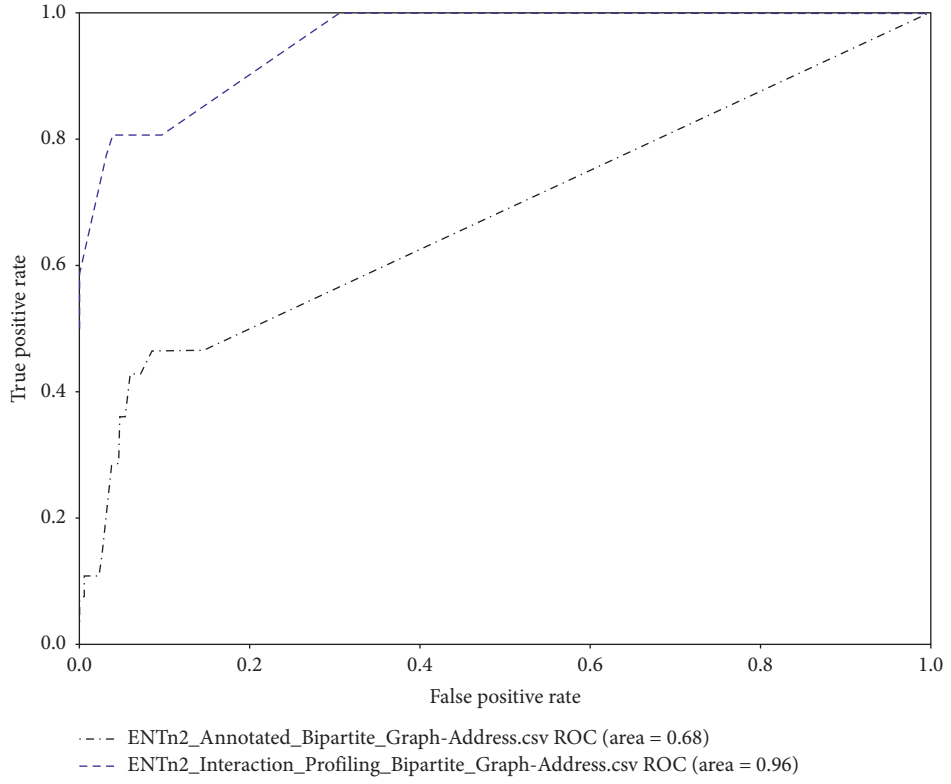

Figure 19: Illustration of the ROC curve for two *Address* feature vectors generated from the two kinds of bipartite graphs for the ENT_{N2} dataset.

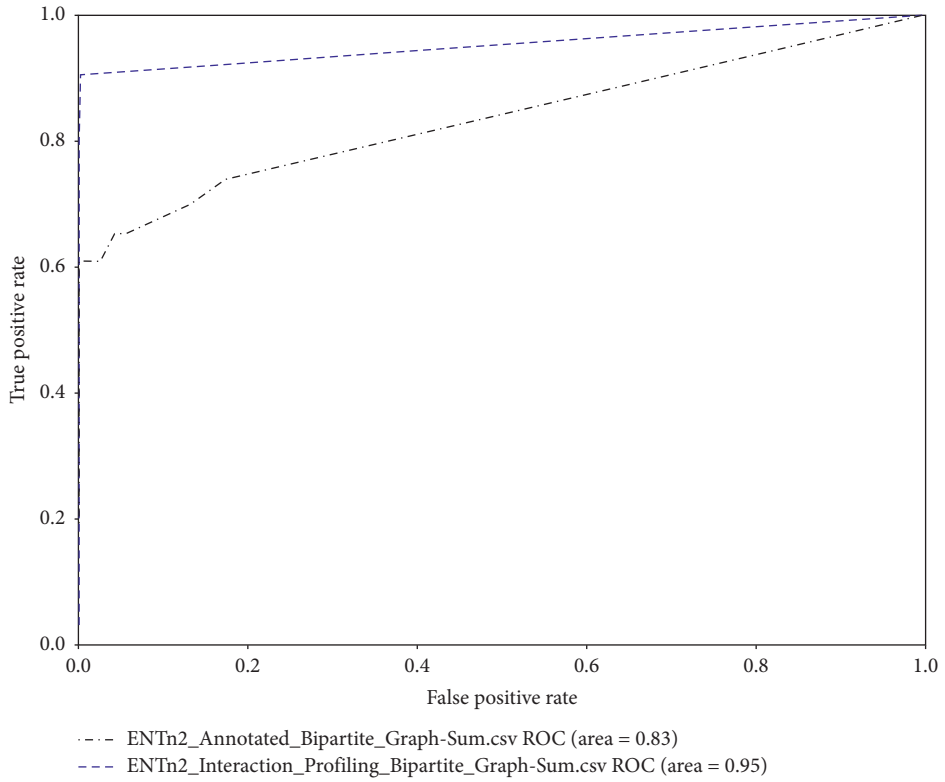

Figure 20: Illustration of the ROC curve of the combined three feature vectors generated from the two bipartite graphs for the ENT_{N2} dataset.

Table 3: Comparison of AUC values generated by annotated bipartite graph and interaction profiling bipartite graph in classification evaluation.

Dataset	Feature vectors	Interaction profiling bipartite graph (AUC) (%)	Annotated bipartite graph (AUC) (%)
ENT_{N1}	Process feature	84.93	85.17
	Address feature	99.97	62.01
	Trace feature	79.61	74.37
	Merged feature	99.97	96.31
ENT_{N2}	Process feature	96.88	79.87
	Address feature	95.86	68.84
	Trace feature	75.29	54.69
	Merged feature	95.34	83.55

Table 4: Top 10 detected malicious domains with higher detected probability.

Domain	Detection probability	Detection rate of VirusTotal
grjxr.snap-affairs.com	0.994728257	0/67
Mdaka.fbhookup.club	0.99454585	0/67
Adserver-g.juicyads.com	0.993425958	0/67
app4.getmacsoft.site	0.991589101	0/67
nofreezingmac.click	0.991547806	2/67
www.por.tw	0.985447549	1/67
extcoolff.com	0.984869145	0/67
urlspirit.spiritsoft.cn	0.981827519	1/67
bak1.spiritsoft.cn	0.981827519	1/67
folder.maroon91.com	0.981491668	0/67

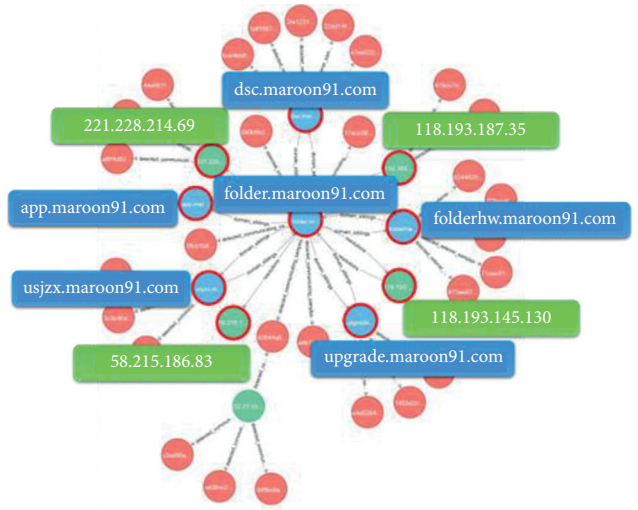

Figure 21: An example of the domain relation graph constructed from external threat intelligence.

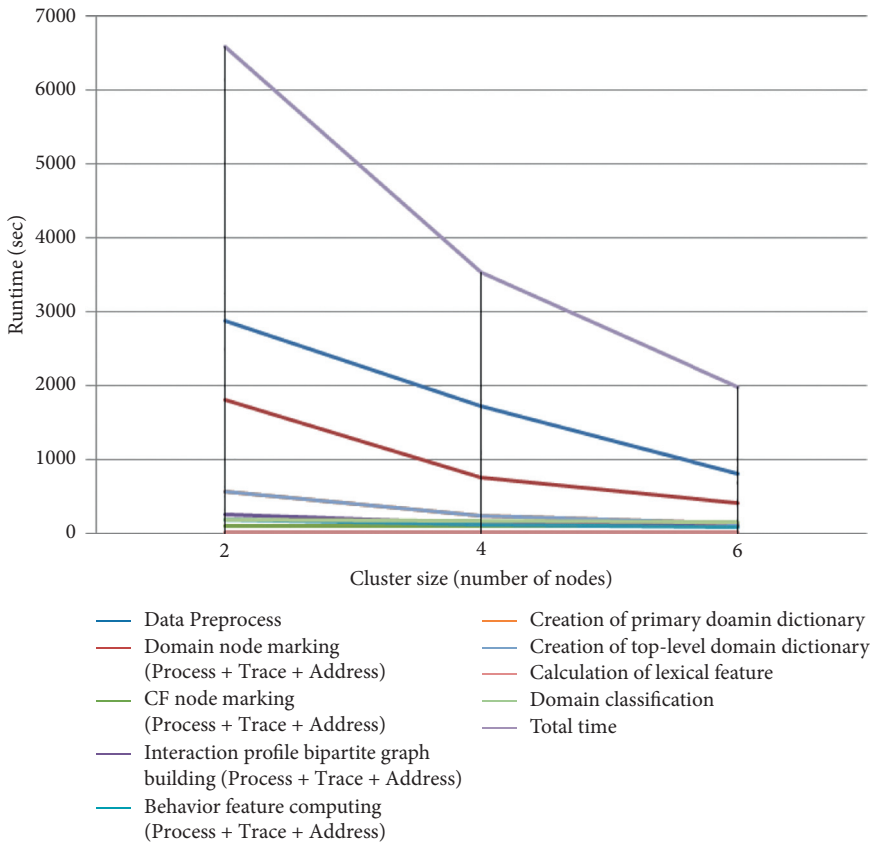

Figure 22: Runtime analysis of the *MD-Miner*^{P}.

graph mining and lexical feature extraction. The *MD-Miner*^{P} designed four MapReduce jobs to accomplish the interaction profiling bipartite graph mining: (1) domain node labeling, (2) *CF* node labeling, (3) interaction profiling bipartite graph building, and (4) behavior feature calculation. The lexical feature extraction can be divided into three MapReduce jobs: (1) creation of primary domain dictionary, (2) creation of top-level domain dictionary, and (3) calculation of lexical features. On the other hand, both the data preprocess stage and domain classification stage have only one MapReduce job. Figure 22 shows the runtime analysis of the *MD-Miner*^{P}. We observe that the data preprocess stage and the domain node labeling of the feature extraction are the primary bottleneck of the *MD-Miner*^{P} process. As the above two jobs

mainly involve I/O operations, the I/O is the primary performance bottleneck in processing the massive data. However, with an increased number of nodes, the computation time of the data preprocess stage and domain node labeling decreases substantially. The experiments show that the *MD-Miner*^{P} tends to possess a superior scalability for the MapReduce.

5. Conclusions

This paper proposes a malicious domain detection system based on a novel bipartite graph called *MD-Miner*^{P}. The interaction profiling bipartite graphs and lexical analysis adopted by the *MD-Miner*^{P} can handle big data. The mining of unknown malicious domains is accomplished by analyzing network interaction behaviors between clients and domains in big network traffic data. The *MD-Miner*^{P} is designed as a scalable system to monitor and analyze big network traffic data to find illegal network activities. Two big network traffic datasets (ENT_{N1} and ENT_{N2}), three validation aspects, and four experiments were proposed to inspect the performance of *MD-Miner*^{P}. The experiments used ROC curves and 10-fold cross-validation with known domains. The experimental results confirm that the feature extraction method proposed by *MD-Miner*^{P} as applied to ENT_{N1} obtained an AUC of 1.00 and applied to the ENT_{N2} obtained an AUC of 0.98. The experimental results of the direct comparison showed that the feature vectors extracted from the interaction profiling bipartite graph are superior to the annotated bipartite graph for both the single and merged feature vectors. In addition, verifying the unknown domain predicted as malicious by the *MD-Miner*^{P} allows the verification method to shape the relationship diagram of the domain. The relationship diagram shows that the domain is directly and indirectly associated with the IP and the domain with malicious behavior. Finally, controlling the number of nodes in the Hadoop cluster verifies that the *MD-Miner*^{P} is a system that fully satisfies the parallel computing conditions, even if the enterprise's network traffic data is large. Therefore, the *MD-Miner*^{P} is applied to conduct malicious domain data mining.

This paper has confirmed the contribution of *MD-Miner*^{P}, but it has some limitations. As described in Section 3, interaction profiling bipartite graph requires domain threat intelligence to label known domain nodes as black and white. Therefore, the quality and quantity of ground truth affect the performance of *MD-Miner*^{P}. Fortunately, collecting public and commercial domain intelligence can effectively overcome this problem. In addition, *MD-Miner*^{P} may not be suitable for DHCP network environment. This is because the proposed bipartite graph uses client-IP to locate individual hosts, and DHCP may cause different hosts to be assigned to the same IP. The solution to this challenge is to correlate DHCP logs with network traffic data to obtain the network behavior of each individual host. The final challenge is that *MD-Miner*^{P} needs to be retrained periodically to maintain detection accuracy. As cybercriminals' technology is constantly evolving, it is necessary to regularly employ *MD-Miner*^{P} and through the latest network traffic data and network threat intelligence to obtain updated domain classification model.

Future work will focus on two areas. First, for detecting malicious domain from big network traffic data, it will be considered whether this approach applies to other large log data, such as firewall and DNS logs. Furthermore, the proposed bipartite graph algorithm can be used to perform correlation analysis for multiple types of network traffic logs to optimize the detection capability. Second, the proposed algorithm is applied to the analysis of other malicious threats. For example, treat the smartphone application's dynamic analysis data (e.g., system call) and static analysis data (e.g., opcode) as *CF*, and match the threat intelligence of applications to build the interaction profiling bipartite graph of applications to mine hidden malicious applications. In addition, the *MD-Miner*^{P} mechanism can be used as the basis for a bilateral market service model [42] to collect malicious traffic. We have provided a website, https://netflowtotal.firebaseapp.com/, to prove this concept [43].

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. Sharif, A. Lanzi, J. Giffin, and W. Lee, "Impeding malware analysis using conditional code obfuscation," in *Proceedings of the 15th An Network and Distributed System Security Symposium (NDSS'08)*, San Diego, CA, USA, Feburary 2008.

[2] J. Oberheide, E. Cooke, and F. Jahanian, "CloudAV: N-version antivirus in the network cloud," in *Proceedings of the 17th An Security Symposium*, pp. 91–106, Boston, MA, USA, July 2008.

[3] A. K. Seewald and W. N. Gansterer, "On the detection and identification of botnets," *Computers & Security*, vol. 29, no. 1, pp. 45–58, 2010.

[4] G. Gu, J. Zhang, W. Lee, and BotSniffer, "Detecting botnet command and control channels in network traffic," in *Proceedings of the 15th An Network and Distributed System Security Symposium (NDSS'08)*, San Diego, CA, USA, Feburary 2008.

[5] 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 An USENIX Security Symposium (USENIX Security '08)*, pp. 139–154, Boston, MA, USA, August 2008.

[6] P. Wurzinger, L. Bilge, T. Holz, J. Goebel, C. Kruegel, and E. Kir-da, "Automatically generating models for botnet detection," in *Proceedings of the 14th An European Symposium on Research in Computer Security (ESOR-ICS'09)*, pp. 232–249, Saint-Malo, France, September 2009.

[7] H. Binsalleeh, T. Ormerod, A. Boukhtouta et al., "On the analysis of the Zeus botnet crimeware toolkit," in *Proceedings of the 2010 Eighth International Conference on Privacy, Security and Trust*, pp. 31–38, Ottawa, Canada, August 2010.

[8] J. Francois, S. Wang, W. Bronzi, R. State, and T. Engel, "Botcloud: Detecting Botnets Using MapReduce," in *Proceedings of the IEEE International Workshop on Information Forensics and Security (WIFS'11)*, pp. 1–6, Delft, The Netherlands, December 2011.

[9] W. T. Strayer, R. Walsh, C. Livadas, and D. Lapsley, "Detecting botnets with tight command and control," in *Proceedings of the 31th An. IEEE Conference on Local Computer Networks*, pp. 195–202, Florida, U.S.A, November 2006.

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

[11] R. R. Panko and J. L. Panko, *Business Data Networks and Security*, pp. 145–147, Pearson, Harlow, Essex, 2015.

[12] J. H. Sun, T. H. Jeng, C. C. Chen, H. C. Huang, and K. S. Chou, "MD-Miner: behavior-based tracking of network traffic for malware-control domain detection," in *Proceedings of the 3rd An IEEE Third International Conference on Big Data Computing Service and Applications (BigDataService'17)*, pp. 96–105, San Francisco, CA, USA, April 2017.

[13] T.-H. Jeng, W.-Y. Luo, C.-C. Huang, C.-C. Chen, K.-H. Chang, and Y.-M. Chen, "Cloud computing for malicious encrypted traffic analysis and collaboration," in *Proceedings of the TANET2018*, IJGHPC, Nanjing, China, 2019.

[14] B. Rahbarinia, R. Perdisci, and M. Antonakakis, "Segugio: efficient behavior-based tracking of malware-control domains in large ISP networks," in *Proceedings of the 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 403–414, Rio de Janeiro, Brazil, June 2015.

[15] S. Y. Chen, T. H. Jeng, C. C. Huang, C. C. Chen, and K. S. Chou, "Doctrina: annotated bipartite graph mining for malware-control domain detection," in *Proceedings of the 7th Ann. International Conference on Communication and Network Security (ICCNS'17)*, pp. 67–75, Tokyo, Japan, November 2017.

[16] R. Holley and D. Rosenfeld, "MAUL: machine agent user learning," 2010, http://cs229.stanford.edu/proj2010/HolleyRosenfeld-MAUL.pdf.

[17] N. Kheir, "Analyzing http user agent anomalies for malware detection," *Data Privacy Management and Autonomous Spontaneous Security,Lecture Notes in Computer Science*, vol. 7731, pp. 187–200, 2013.

[18] J. P. John, A. Moshchuk, S. D. Gribble, and A. Krishnamurthy, "Studying spamming botnets using botlab," in *Proceedings of the 6th Ann. USENIX Symposium on Networked Systems Design and Implementation (NSDI'09)*, vol. 9, pp. 291–306, Boston, MA, USA, April 2009.

[19] T. H. Jeng, Y. M. Chen, C. C. Chen, C. C. Huang, and K. S. Chou, "Interaction profiling bipartite graph mining for malicious network activity detection," in *Proceedings of the IEEE Conference on Dependable and Secure Computing (DSC 2018)*, pp. 323–330, Kaohsiung, Taiwan, December 2018.

[20] S.-T. Liu, Y.-M. Chen, and H.-C. Hung, "N-victims: an approach to determine N-victims for APT investigations," *Information Security Applications*, vol. 7690, pp. 226–240, 2012.

[21] En.wikipedia.org. 2019. Advanced Persistent Threat. https://en.wikipedia.org/wiki/Advanced_persistent_threat.

[22] C. Tankard, "Advanced persistent threats and how to monitor and deter them," *Network Security*, vol. 2011, no. 8, pp. 16–19, 2011.

[23] D. Alperovitch, "Revealed: operation shady RAT," 2011, http://www.csri.info/wp-content/uploads/2012/08/wp-operation-shady-rat1.pdf.

[24] K. F. Hong, C. C. Chen, Y. T. Chiu, and K. S. Chou, "Ctracer: uncover C&C in advanced persistent threats based on scalable framework for enterprise log data," in *Proceedings of the IEEE International Congress on Big Data*, pp. 551–558, New York, NY, USA, June 2015.

[25] K. F. Hong, C. C. Chen, T. T. Chiu, and K. S. Chou, "Scalable command and control detection in log data through UF-ICF analysis," in *Proceedings of the International Carnahan Conference On Security Technology*, pp. 293–298, Quebec, Canada, October 2015.

[26] R. Perdisci, W. Lee, and N. Feamster, "Behavioral clustering of http-based malware and signature generation using malicious network traces," in *Proceedings of the 7th An USENIX Conference on Networked Systems Design and Implementation (NSDI'10)*, pp. 391–404, San Jose, CA, USA, April 2010.

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

[28] S. Marchal, J. Francois, R. State, and T. Engel, "PhishStorm: detecting phishing with streaming analytics," *IEEE Transactions on Network and Service Management*, IEEE Transactions), vol. 11, no. 4, pp. 458–471, 2014.

[29] 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 12th Ann. International Conference on Privacy, Security and Trust (PST'14)*, pp. 249–256, Toronto, Canada, July 2014.

[30] D. Chiba, T. Yagi, M. Akiyama, K. Aoki, T. Hariu, and S. Goto, *BotProfiler: Profiling Variability of Substrings in Http Requests to Detect Malware-Infected Hosts*, IEEE Trust-com/BigDataSE/ISPA, Piscataway, NJ ,USA, pp. 758–765, 2015.

[31] S. Garera, N. Provos, M. Chew, and A. D. Rubin, "A framework for detection and measurement of phishing attacks," in *Proceedings of the ACM Workshop on Recurring Malcode (WORM'07)*, pp. 1–8, Alexandria, VA, USA, November 2007.

[32] A. Ramachandran and N. Feamster, "Understanding the network-level behavior of spammers," in *Proceedings of the Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'06)*, pp. 291–302, London, UK, September 2006.

[33] S. Hao, N. A. Syed, N. Feamster, A. G. Gray, and S. Krasser, "Detecting spammers with SNARE: spatio-temporal network-level automatic reputation engine," in *Proceedings of the 18th An USENIX Security Symposium (SSYM'09)*, pp. 101–118, Montreal, Canada, August 2009.

[34] M. P. Collins, T. J. Shimeall, S. Faber et al., "Using uncleanliness to predict future botnet addresses," in *Proceedings of the 7th Ann. ACM SIGCOMM Conference on Internet measurement (IMC'07)*, pp. 93–104, San Diego, CA, USA, October 2007.

[35] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a dynamic reputation system for DNS," in *Proceedings of the 19th An. USENIX Conference on Security (USENIX Se-Curity'10)*, p. 18, Washington, DC, USA, August 2010.

[36] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "Exposure: finding malicious domains using passive dns analysis," in *Proceedings of the 18th Ann. Network and Distributed System*

Security Symposium (NDSS'11), San Diego, CA, USA, Feburary. 2011.

[37] B. Liang, J. Huang, F. Liu, D. Wang, D. Dong, and Z. Liang, "Malicious web pages detection based on abnormal visibility recognition," in *Proceedings of the E-Business and Information System Security (EBISS'09)*, pp. 1–5, Wuhan, China, May 2009.

[38] D. Sahoo, C. Liu, and S. C. H. Hoi, "Malicious url detection using machine learning: a survey," 2017, http://arxiv.org/abs/1701.07179.

[39] P. Kolari, T. Finin, and A. Joshi, "SVMs for the blogosphere: blog identification and splog detection," in *Proceedings of the AAAI Spring Symposium on Computational Approaches to Analyzing Weblogs*, pp. 92–99, Franco, Egypt, March 2006.

[40] E. Cooke, F. Jahanian, and D. McPherson, "The zombie roundup: understanding, detecting, and disrupting botnets," in *Proceedings of the First Ann. Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI'05)*, pp. 39–44, Cambridge, MA, USA, July 2005.

[41] B. Fish, "On the computational complexity of mapreduce," in *Proceedings of the International Symposium on Distributed Computing*, Delft, The Netherlands, October 2015.

[42] T.-H. Jeng, W.-M. Chan, W.-Y. Luo et al., "A cloud service integration platform for malicious traffic analysis and collaboration," in *Proceedings of the ICCBD*, Taichung, Taiwan, October 2019.

[43] T. Hastie, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, pp. 316-317, Springer, New York, NY, USA, 2009.