electronics

Article

A Word-Level Analytical Approach for Identifying Malicious Domain Names Caused by Dictionary-Based DGA Malware

Akihiro Satoh [1,*], Yutaka Fukuda [1], Gen Kitagata [2] and Yutaka Nakamura [1]

1 Information Science and Technology Center, Kyushu Institute of Technology, Kitakyushu 804-8550, Japan; fukuda@isc.kyutech.ac.jp (Y.F.); yutaka-n@isc.kyutech.ac.jp (Y.N.)
2 Research Institute of Electrical Communication, Tohoku University, Sendai 980-8577, Japan; minatsu@riec.tohoku.ac.jp
* Correspondence: satoh@isc.kyutech.ac.jp; Tel.: +81-093-884-3000

Abstract: Computer networks are facing serious threats from the emergence of malware with sophisticated DGAs (Domain Generation Algorithms). This type of DGA malware dynamically generates domain names by concatenating words from dictionaries for evading detection. In this paper, we propose an approach for identifying the callback communications of such dictionary-based DGA malware by analyzing their domain names at the word level. This approach is based on the following observations: These malware families use their own dictionaries and algorithms to generate domain names, and accordingly, the word usages of malware-generated domains are distinctly different from those of human-generated domains. Our evaluation indicates that the proposed approach is capable of achieving accuracy, recall, and precision as high as 0.9989, 0.9977, and 0.9869, respectively, when used with labeled datasets. We also clarify the functional differences between our approach and other published methods via qualitative comparisons. Taken together, these results suggest that malware-infected machines can be identified and removed from networks using DNS queries for detected malicious domain names as triggers. Our approach contributes to dramatically improving network security by providing a technique to address various types of malware encroachment.

Keywords: malware detection; dictionary-based domain generation algorithm; domain name; network security

check for updates

Citation: Satoh, A.; Fukuda, Y.; Kitagata, G.; Nakamura, Y. A Word-Level Analytical Approach for Identifying Malicious Domain Names Caused by Dictionary-Based DGA Malware. *Electronics* **2021**, *10*, 1039. https://doi.org/10.3390/electronics10091039

Academic Editors: Jemal H. Abawajy, Aaqif Afzaal Abbasi and Amir Sinaeepourfard

Received: 29 March 2021
Accepted: 26 April 2021
Published: 28 April 2021

1. Introduction

Some of the most serious security threats currently faced by computer networks involve malware. Most commonly, cybercriminals take control of malware-infected computers through a command-and-control server (C&C) and then use those computers to engage in malicious activities such as stealing confidential information, spreading malware to additional computers, and phishing within an organization. According to a recent McAfee report [1], over 300,000 new forms of malware and their variants are created each day, and the global annual cost of malware-related cybercrime may be as high as $600 billion. Therefore, the establishment of security mechanisms to protect networks against malware encroachment is imperative.

To prevent malware-related damage, administrators must rapidly identify and remove any infected computers that may reside in their networks. However, many malware families use domain generation algorithms (DGAs) to avoid detection [2]. A DGA is a technique in which the domain name is changed frequently to hide the callback communication from the infected computer to the C&C. Specifically, DGA malware dynamically generates and then attempts to resolve domain names. The domain name that returns the correct response is considered as the C&C. However, since the lifetimes of domain names generated for such callbacks are extremely short, it is difficult for conventional security appliances that monitor communications with known malicious domains to detect callbacks caused by DGA malware.

Some previous studies [3–5] have focused on identifying discernible differences in the character strings of benign and malicious domain names for detecting the callbacks of DGA malware. For example, Truong et al. [4] proposed a method that learns and predicts the character patterns in domain names using bigram models with supervised learning algorithms, and Anderson et al. [5] extended this method to include character-level modeling with long short-term memory (LSTM) networks. These methods are based on two observations: Benign domain names tend to reflect the intent of their domain registrants, such as an organization, product, or content, while malicious domain names tend to be meaningless character strings because conflicts with already registered domain names must be avoided. These efforts have spurred the emergence of more sophisticated DGA malware to defeat these detection methods, including new types of malware families that have their own dictionaries from which they can concatenate words to dynamically generate domain names that are difficult to distinguish from human-generated domain names [6]. As a result, character-level modeling, which excludes words, is unlikely to be sufficiently accurate to detect such dictionary-based DGA (dict-DGA) malware.

In this paper, we aim to identify dynamically generated domains from massive domain name system (DNS) queries to detect the callbacks of dict-DGA malware. We focus on queried domain names for the traditional DNSs, in which name resolution is an unencrypted interaction that occurs prior to data exchange between malware and C&C. In fact, numerous studies have been reported to detect various malicious activities by analyzing queries for the traditional DNSs [7]. The main issue is character-level similarities between dynamically generated domain names and human-generated domain names. Therefore, we propose an approach for identifying malicious domain names by analyzing their character strings at the word level. This approach is based on the following observations: Each type of dict-DGA malware uses its own dictionaries and algorithms to generate domain names, and the word usages of malware-generated domains are distinctly different from those of human-generated domains. Note that natural language processing techniques cannot be effectively used to the analysis of domain names because domain names are short, unstructured, and unsegmented character strings. Here, an unstructured character string means that each word has no role such as subject, verb, object, or predicate. Chen et al. [8] designed a model with knowledge powered attention mechanisms for classifying short texts according to their semantics. A further example of short text analysis is spam identification in social media posts, emails, and short messaging services [9,10]. However, typical short texts are structured and segmented character strings such as "Jay and Jolin are born in Taiwan" in [8] while domain names are unstructured and unsegmented character strings such as `incometaxindiaefiling.gov.in`, `teacherspayteachers.com`, and `adnetworkperformance.com` in the top one million sites provided by Alexa [11]. Moreover, domain names have no contextual information that typical short texts have.

The remainder of this paper is organized as follows. In Section 2, we review related studies and discuss their limitations. In Section 3, as a way to remove malware-infected computers, we propose an approach to identify malicious domain names by analyzing their character strings at the word level. Then, Section 4 describes the experiments conducted to analyze the effectiveness of our proposed approach for detecting the callbacks of dict-DGA malware. Finally, we summarize our conclusions and directions for future work in Section 5.

2. Background

Because the interconnectivity and reachability of computer networks make them vulnerable to malware hazards, numerous research communities have expressed strong ongoing interest in the development of adequate defense solutions [12–14]. Accordingly, we begin this section by describing the details of DGA malware, after which we introduce studies aimed at detecting various malware and note their limitations.

2.1. DGA Malware

Malware programs such as Conficker and Kraken, which have caused considerable worldwide damage, implement DGAs as one aspect of their capabilities. Moreover, researchers have found improper codes embedded in websites and web-based advertisements that are designed to promote the spread of such malware programs over broad target areas [15,16].

Figure 1 schematically depicts callbacks implemented by DGA malware. In this figure, communications labeled **Q** are requests for name resolution from DGA malware to a recursive DNS server (RDNS), while communications labeled **R** are responses to these requests. We begin by assuming that DGA-generated domain names, being the capability of a C&C, are preregistered in an authoritative DNS server (ADNS). The malware first dynamically generates multiple domain names, such as `d8wgr9gpa7.com`, `fpeqbiwxfx.edu`, and `gxwx123nrs.net`, based on its DGA and then directs DNS queries regarding these domain names to the RDNS in the network to which the malware itself belongs. Upon receiving a query, the RDNS returns the address assigned to the domain name if it is registered in the ADNSs or a nonexisting domain (NXDOMAIN) response if it is not. Finally, any domain, e.g., `gxwx123nrs.net` in this figure, that corresponds to an affirmative DNS response is assumed to be the C&C, and the malware attempts callbacks for the address assigned to that domain.

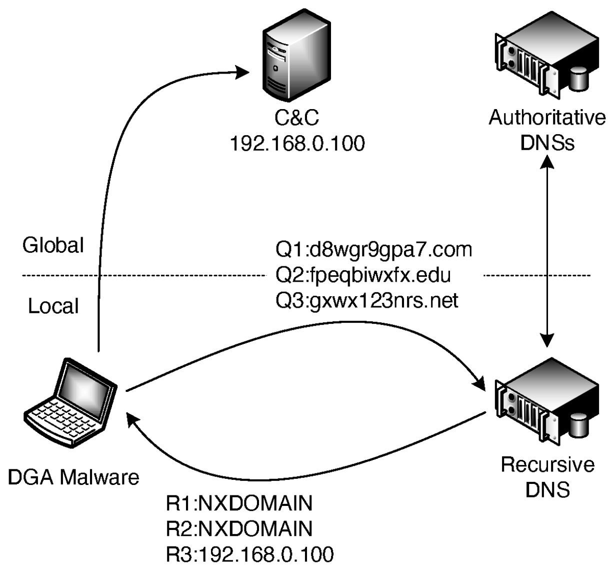

Figure 1. Callback communication between DGA malware and C&C.

The character strings generated by typical DGA malware for use as malicious domain names are easily distinguished from benign domain names. In GameOver Zeus [17], domain names are random sequences of letters and numerals ranging in length from 15 to 30 characters, such as `1ygx14u1vnf8hb1twhv8619h8ygr.net` and `cipu0wdgsnq9u8s t8m1lym0hq.com`. These domain names are designed to avoid namespace collisions with previously registered domains. In Rovnix [6], a type of dict-DGA malware, domain names are generated by concatenating words from dictionaries, such as `accelerateaccountant .in.net` and `accelerateactress.in.net`. These domain names are difficult to distinguish from human-generated domain names due to their string similarities.

The objective of a DGA is to establish highly available communication channels between the malware and the C&C. One result of this process is that communication barriers based on blacklists are easily avoided by changing C&C domains. Furthermore, communications from within the network to the outside span a wide range of destination addresses and are thus difficult to detect since they are not restricted by firewalls. Notably, with identical DGAs, the malware and the C&C avoid the need to exchange any information related to changing the domain name.

2.2. Related Work

Studies that focus on producing more sophisticated blacklists are conducted frequently and constitute the core of most network threat defense strategies. For example, Soldo et al. [18] proposed a method to substantially enhance the performance of blacklists based on previous attack logs provided by multiple contributors. Meanwhile, Freudiger et al. [19] improved the confidentiality of this method by sharing attack logs through peer-to-peer communications. Separately, Špaček et al. [20,21] developed a DNS firewall system that blocks communications from a protected network to malicious domains on outside networks. This system uses DNS response policy zone (RPZ) technology [22] for advanced domain blacklisting. Unfortunately, malware families constructed using DGAs are capable of avoiding blacklist-based detection by frequently changing the domain of their C&C.

In another approach, deep packet inspection (DPI), which surveys packet payloads, has been used to strengthen protections against malware encroachments. For example, Gu et al. [23] implemented BotHunter, which is a DPI-based passive network monitoring system that models typical malware behavior and interprets communications exhibiting strong associations with such behavior as evidence of malware contamination. Other studies have discussed efforts to improve the performance of DPI-based detection [24,25]. However, in a recent cybersecurity report [26], Cisco Systems noted that more than 50% of current Internet communications are encrypted and that approximately 70% of all malware programs encrypt their communications. Thus, in inverse proportion to the fraction of encrypted communications, the share of future communications amenable to DPI analysis will eventually be reduced to a negligible subset. A common practice in applying DPI to encrypted packets is by employing a Man-in-the-Middle approach [27], which interrupts, decrypts, and analyzes communications between two end-points, but this approach has both practical and privacy concerns. Specifically, the large latency is incurred due to computational costs in encrypt-decrypt processes, and the privacy guarantee is violated, raising potential compliance issues. These scenarios have motivated focus on the resolution of domain names for the traditional DNSs, which are not encryption, as an information source for detecting malware. Through previous investigations [28], we have confirmed that the traces of malicious activities appear in queries for the traditional DNSs.

Rahbarinia et al. [29] developed a system, called Segugio, that finds unknown malicious domains based on their co-occurrence relation with known malicious domains in DNS queries. Segugio is based on the following insights: (1) Infected computers in the same malware family tend to communicate with the same malicious domain group and (2) uninfected computers have no reason to communicate with malicious domains. However, since the temporary malicious domains used for DGA malware callbacks have extremely short lifetimes and since the domains that co-occur with temporary malicious domains may not actually exist, this system is not sufficiently able to detect the callbacks of DGA malware.

In another example, Berger et al. [30] developed a system called DNSMap that discovers potentially compromised computers based on DNS traffic. Specifically, DNSMap identifies suspicious agile DNS mappings, i.e., mappings characterized by rapidly changing domain names and/or addresses, which are often used by the C&C. Meanwhile, Wang et al. [31] deployed a system called DBod, which classifies and detects DGA malware based on the analytical results of DNS query behaviors. Moreover, since computers contaminated by the same DGA malware family tend to generate a large number of identical DNS queries, those queries also tend to exhibit a similar domain scope and distribution. Thus, DBod exploits these similarities for classification and detection. However, since both DNSMap and DBod require the observation of an enormous quantity of extensive DNS traffic, their utility is limited to large-scale networks, such as Internet service providers (ISPs).

Plohmann et al. [32] revealed the DGA landscape by reverse-engineering 43 malware families, and Zago et al. [33] provided a mature dataset with analysis results for over 30 million domains generated by 50 malware families. Based on those results, other researchers have attempted to distinguish between benign and malicious domains using only their character strings in manners similar to our approach. For example, Truong et al. [4] proposed a technique that learns and predicts character patterns using bigram models with supervised learning algorithms, and Anderson et al. [5] extended this technique using character-level models with LSTM networks. Meanwhile, Qiao et al. [34] combined LSTM networks with attention mechanisms to assign appropriate weight values to the characters in domain names. In addition, Vinayakumar et al. [35] compared the performance of various machine learning (ML) and deep learning (DL) algorithms when detecting DGA-generated malicious domains. These techniques are based on the existence of discernible bias in the rules for generating malicious domain names and thus must learn the bias in advance by analyzing both benign and malicious datasets.

Pereira et al. [36] proposed a method for detecting dict-DGA malware based solely on domain strings. This method has two functions: Estimating the dictionary used by dict-DGA malware from their callbacks and identifying malicious domain names using the estimated dictionary. However, this method is extremely simple in that malicious domain names are identified by whether the number of words constituting the character string exceeds a given threshold in the estimated dictionary. As a result, numerous false positives are caused by the inability to consider benign domain names. In contrast, our work focuses on using ML to characterize the differences between benign and malicious domain names and thus produces better results than can be achieved by the method in [36].

3. Proposal

In this paper, we report on attempts to detect dict-DGA malware by identifying malicious domain names from massive DNS queries. We focus on queried domain names for the traditional DNSs, in which name resolution is an unencrypted interaction that occurs prior to data exchange between malware and C&C. Table 1 shows examples of domain names generated by dict-DGA malware. Because these domains depend on the C&C of the callback destination, each type of malware uses its own dictionaries and algorithms to generate them. For example, a comparison of the Gozi and Matsnu dictionaries reported in Refs. [37,38] reveals that the dictionaries contain 975 and 1391 words, respectively, with only 240 words appearing in both dictionaries (Ursnif in literature [37] is an alias name for Gozi). Moreover, the word usage is expected to exhibit significant differences between malware-generated domain names and human-generated domain names. Based on these observations, we propose an approach for identifying malicious domain names by analyzing character strings at the word-level. The novelties of this approach include the following: (1) the use of general graph-theoretical techniques to represent relationships among words in domain names; and (2) the use of centrality to quantify the importance of each word in this graph. These indicators characterize the differences between benign and malicious domain names on the basis of word biases in their character strings. Finally,

Table 1. Examples of domain names generated by dict-DGA malware.

Banjori	`earnestnessbiophysicalohax.com` `pbmnestnessbiophysicalohax.com`
Gozi	`williamseasily.com` `printingthatlabel.com`
Matsnu	`shoulderracerecognizeblue.com` `emergencyadaptselectdoubt.com`
Suppobox	`windowtherefore.net` `severadifference.net`

we apply ML techniques to the feature vectors derived from these indicators and thereby reduce the number of false positives in identification.

Figure 2 shows an overview of the proposed approach, which has four steps: (1) Noise reduction, (2) word segmentation, (3) word graph construction, and (4) feature-vector-based identification. The following sections describe the training datasets and each of these steps in detail.

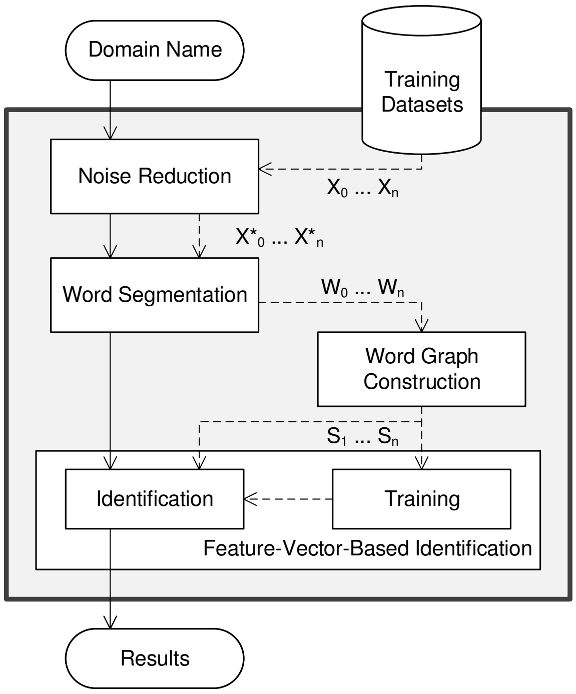

Figure 2. Overview of our proposed approach for identifying malicious domain names by analyzing their character strings at the word level.

3.1. Training Datasets

We denote a dataset comprising the same type of domains by $x \in X_i, i \in 0 \cdots n$, where the dataset X_0 contains benign domains and the remaining n datasets $X_1, \cdots, X_n$ contain malicious domains generated by n different malware families. In other words, malicious domains are classified into the datasets of each malware family. Note that domain names in the datasets are shortened and replaced with primary domain names. A primary domain is the highest-level domain name given to a registrar [39]. For example, the primary domain names for `www.mdpi.com`, `mail.tohoku.ac.jp`, and `ns.isc.kyutech.edu` would be `mdpi.com`, `tohoku.ac.jp`, and `kyutech.edu`, respectively.

As noted in Section 2.1, a change in a callback destination by DGA malware is commonly accompanied by NXDOMAIN messages as the result of forward lookup. In such cases, the NXDOMAIN response is an error message that indicates an invalid domain name. Based on this insight, our approach focuses attention on specific domains for which the name resolution meets both the forward lookup request and NXDOMAIN message response.

3.2. Noise Reduction

The first step is to identify benign domains via comparison with whitelists, which consist of domains owned by trustworthy organizations and those used by specific services that cause massive NXDOMAIN messages. Examples of these domains include `kyutech.ac.jp` and `kyutech.jp`, which are owned by our institution; `uribl.com` and `dnswl.org`, which are queried for DNS-based lists (DNSBL) [40]; and `trendmicro.com`, which is associated with endpoint antivirus software. Next, domain names that violate DNS specifications are deemed to be the result of misconfiguration and are discarded [41].

Notably, any domains prefixed with "xn--" are excluded here because our approach does not support internationalized domain names (IDNs) [42]. We expect IDNs to be not often used by dict-DGA malware that generates domain names from word dictionaries because in name resolution IDNs are replaced by seemingly random character strings, such as `xn--sjqw6xkwbgyd9ay88l` and `xn--djry4lk1bj6u`, which are distinctly different from human-generated domain names. Incidentally, one of the most common abuses of IDNs is the registration of domain names that exhibit a high visual and semantic similarity to those of reputable brands [43,44]. The remaining domain names are considered possible DGA-generated strings and are passed as training datasets $X_i^*, i \in 0 \cdots n$ to subsequent functional units for further analysis.

3.3. Word Segmentation

This step segments the primary-level character string for domain x into a word group w using $\mathbb{D}$, which consists of an English dictionary and a corpus collected through web crawling. Word segmentation is based on two conditions: (1) Minimizing the number of words while maximizing word lengths, and (2) giving preference to words that are in the dictionary by increasing their likelihood of being selected.

We define this word segmentation approach with the following equation:

$$\mathcal{F}(x) = \underset{w \in \mathbb{W}(x)}{\arg\max} \frac{1}{m} \prod_{j=1}^{m} \mathcal{P}(w_j)$$
$$\mathcal{P}(w_j) = \begin{cases} 1 & (w_j \in \mathbb{D}) \\ 1/|\mathbb{D}|^{|w_j|} & (w_j \notin \mathbb{D}) \end{cases}$$

Here, $\mathbb{W}(x)$ is all the candidate segmentations for the primary-level character strings of domain x; w is the candidate segmentation comprising words $w_1, \cdots, w_j, \cdots, w_n$; $|w_j|$ is the length of word w_j; and $|\mathbb{D}|$ is the total number of words in dictionary $\mathbb{D}$. Furthermore, $\mathcal{P}(w_j)$ is the selectivity of word w_j, which is based on whether word w_j is in dictionary $\mathbb{D}$. $\mathcal{P}(w_j) = 1$ when $w_j \in \mathbb{D}$, and $\mathcal{P}(w_j) = 1/|\mathbb{D}|^{|w_j|}$ when $w_j \notin \mathbb{D}$. As an example of word segmentation, the primary-level character string `kyutechlocaldomain` is divided into the word group {`kyutech`, `local`, `domain`}. When this algorithm step is complete, a set of word groups W_i derived by segmenting the primary-level character strings is output for all the domains in training dataset X_i^*. Note that set W_i can contain duplicate elements.

3.4. Word Graph Construction

This step uses a word graph to represent the relationships among a set of word groups W_i corresponding to domain strings in training dataset X_i^*. A word graph G_i is simply a weighted undirected graph with a vertex set and an edge set. Each vertex corresponds to a word in set W_i. The edge weights indicate the frequencies of co-occurrence of words in set W_i, where co-occurrence means that the words belong to the same word group. Note that words with similar character strings share the same vertex. We use the Levenshtein ratio for similarity calculations and unweighted centroid clustering with threshold th_α for word aggregation. Figure 3 shows an example of a constructed word graph using the set of three-word groups [{`kyutech`, `local`, `domain`}, {`kyutech`, `local`, `net`}, {`mdpi`, `domains` }]. Due to the similarity of `domain` and `domains`, the two words share a vertex.

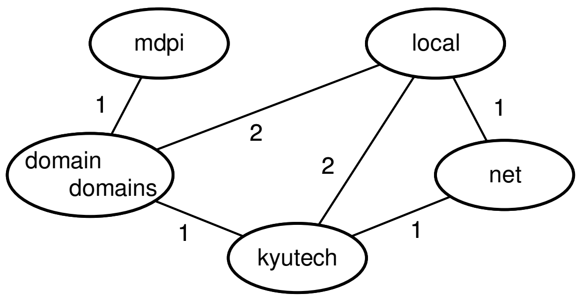

Figure 3. Example of constructing a word graph using a set of word groups.

Dynamically generated malicious domain names and human-generated benign domain names exhibit clear differences in their word co-occurrence frequencies, and the words that illuminate these distinctions most effectively play the central role in a word graph. Specifically, words that have low centrality in word graph G_0, which is constructed from benign dataset X_0, and that have high centrality in another word graph are effective for identifying malicious domains. Based on these findings, we first define conjunction as a process that uses two-word graphs. Specifically, conjunction is a process of reconstructing word graph $G_{i:j}$ from word group sets W_i and W_j into word graphs G_i and G_j.

Figure 4 shows an example of the conjunction process $G_{i:j}$ for word graphs G_i and G_j. Here, the vertices in white represent words in word graph G_i, the vertices in black represent words in word graph G_j, and the vertices in gray represent words that are common to both. Note that the weights of edges are updated with the conjunction process $G_{i:j}$. We then associate the following importance value with an arbitrary group w comprising words $w_1, \cdots, w_j, \cdots, w_n$:

$$\mathcal{S}_i(w) = \sum_{w_j \in w} |w_j| \big(\mathcal{C}_{0:i}(w_j) - \mathcal{C}_0(w_j) \big)$$

Here, $|w_j|$ denotes the length of word w_j. $\mathcal{C}_0(w_j)$ and $\mathcal{C}_{0:i}(w_j)$ are functions that derive the centrality of word w_j in word graphs G_0 and $G_{0:i}$. Accordingly, this value means that the change in the centrality of word group w resulting from the conjunction of word graph G_0 is constructed from benign dataset X_0 and that G_i is constructed from malicious dataset X_i. To compute the values, we use the PageRank centrality, which can be used even if a word graph is undirected and disconnected.

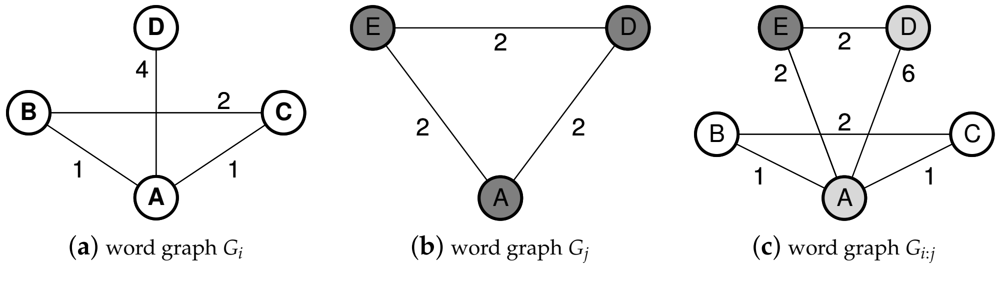

Figure 4. Example of the conjunction process $G_{i:j}$ for word graphs G_i and G_j.

3.5. Feature-Vector-Based Identification

This step first calculates feature vectors from the domains in the training datasets. Next, based on the importance values in the word graph defined in the previous section, the feature vector for the domain name consisting of word group w is given by the following equation:

$$\vec{w} = \big(\mathcal{S}_1(w), \cdots \mathcal{S}_i(w), \cdots \mathcal{S}_n(w) \big)$$

Then, we apply ML to these feature vectors to construct a training model. Specifically, we adopt support vector machine (SVM) as the ML algorithm because of its generalization and identification performance. The output of this step is the overall identification results of our algorithm regarding the benign versus malicious nature of the unknown input domain names, which are obtained by reducing noise, segmenting the domain name, calculating the feature vector, and applying the training model.

4. Evaluation

This section presents evidence to support the effectiveness of the proposed approach in detecting callbacks instigated by DGA malware. We begin by verifying the identification accuracy of this approach from various perspectives and then extensively discuss its effectiveness and limitations. The experimental setup and results are described in Sections 4.1 and 4.2. Finally, we clarify the functional differences between our approach and other published methods via qualitative comparisons in Section 4.3.

4.1. Experimental Setup

Figure 5 shows the layout of our campus network, which has two class B address blocks. This network consists of one core network, 135 access networks managed by 30 departments, and three wireless networks. We manage only the core and wireless networks, including the connection points to the access networks. The total number of media access control (MAC) addresses observed at their connection points was approximately 9500. A total of 470 access points are located in the three wireless networks that geographically cover most of our campus. The computers of more than 6000 employees and students, in addition to numerous visitors, routinely connect to the three wireless networks. The total number of computers connected to the three wireless networks is approximately 6500 at any given point in time. Broken down by operating system (OS), 43% of those computers use iOS, 28% use Windows, 22% use Android, and 7% use macOS. As expected, there are no details regarding the 135 access networks outside our management area. The dataset used for the experiments comprises DNS queries observed on three RDNSs in the campus network during the one-month period beginning on 1 August 2018, with a total size of approximately 65 GB.5 Note that these three RDNSs were used on our entire campus network, including the access networks outside our management areas. The results show that 3,021,124 queries, i.e., approximately 15% of all queries, meet both a forward lookup request and an NXDOMAIN message response.

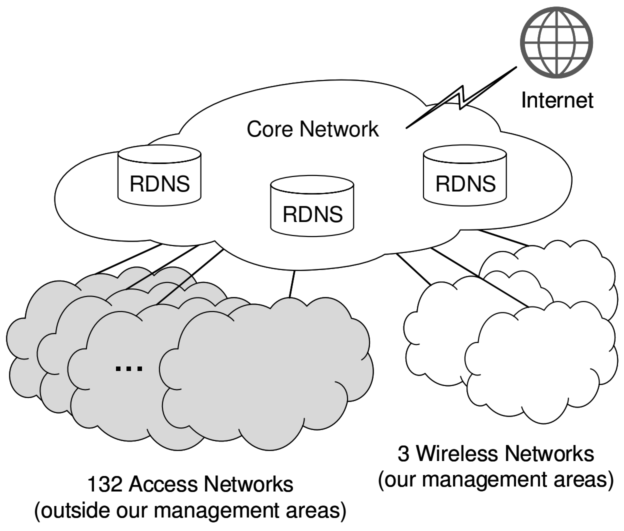

Figure 5. Layout of our campus network.

Table 2 shows the datasets of the benign and malicious domains used in the experiments. The benign domains X_0 were the 3,021,124 queries, i.e., approximately 15% of all queries, collected via the campus network described above. We confirmed the absence of malware-related domains in the benign set via the following procedure. First, suspicious computers were detected by multiple security appliances, after which they were examined for matches with the characteristics reported in [6,31,32], which reported that a DGA malware will generate massive numbers of NXDOMAIN responses as a result of name resolution for unfamiliar domains during a short period. Notably, even though this approach cannot guarantee the absence of malicious domains with perfect certainty, the number of malicious domains likely to remain among the set of benign domains, if not zero, was considered to be extremely small and unlikely to have a significant impact on our experimental results. In contrast, the malicious domains of seven DGAs $X_i, i \in 1 \cdots 7$ were published on the websites [45,46]. Banjori X_1 and Sisron X_6 dynamically generated domain names by combining random characters with specifically selected words, while the other types dynamically generated domain names by concatenating words in their own dictionaries. Banjori and Sisron are not strictly categorized as dict-DGA malware, but we included them in the datasets because they use meaningful words as domain components. In our experiments, five-fold cross-validation of the benign and malicious domains was performed using 20% of the data as the testing datasets and the remainder as the training datasets.

Table 2. Numbers of benign and malicious domains in the datasets.

X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7
Benign	Banjori	Gozi	Matsnu	Pizd	Rovnix	Sisron	Suppobox
3,021,124	30,000	30,000	30,000	30,000	30,000	30,000	30,000

For comparison with our approach, we implemented two different methods for identifying benign and malicious domains based solely on their character strings, as described in [5,36]. The first implementation used character-level modeling with LSTM networks applied to the domain strings, whereas the second implementation used the number of words that constitute the domain string and are present in the estimated dictionary as the identification threshold. Table 3 shows the specifications of the computer used for implementation and calculation, and Table 4 shows the calculation time in our approach and the two implementations required to train 2,584,900 benign and malicious domains, which is 80% of the datasets, and to predict 646,224 benign and malicious domains, which is 20% of the datasets. Here, this computer is equipped with a dedicated GPU to accelerate the training phase in the first implementation.

In our proposed approach, we mainly used words registered in Aspell [47] and the corpus in [48] as the dictionary for word segmentation. The total number of words in the dictionary was 500,000. We set the threshold th_α to 0.85 and the SVM kernel to a radial basis function with hyperparameter 1.0 and cost 5.0. These parameters were determined experimentally, and their optimization will be addressed in future work.

Table 3. Specifications of the computer used for implementation and calculation.

CPU	Xeon Silver 4110 (8core 2.10 GHz)
GPU	NVIDIA GeForce RTX 2080 Ti (11 GB GDDR6)
RAM	96 GB DDR4-2666
SSD	Seq. Read and Write up to 560 MB/s and 530 MB/s
Kernel	Linux 3.10.0-957.1.3.el7.x86_64
Software	TensorFlow 2.0.0, CUDA Toolkit 10.0, cuDNN 7.6.5

Table 4. Calculation time for each phase in our approach and two implementations.

	Training	**Predicting**
Anderson et al. [5]	10,860 s	402 s
Pereira et al. [36]	118 s	72 s
Our work	82,618 s	2896 s

4.2. Experimental Results and Discussion

We employ three common metrics to characterize the ability to identify benign and malicious domains. The first is accuracy, which is defined as the ratio of the number of correctly predicted benign and malicious domains to the total number of benign and malicious domains; the second is recall, which is defined as the ratio of the number of correctly predicted malicious domains to the total number of actual malicious domains; and the third is precision, which is defined as the ratio of the number of correctly predicted malicious domains to the total number of predicted malicious domains.

The experimental results are presented in Table 5, where each value is the mean of the five-fold cross-validation results. The proposed approach achieves an accuracy of 0.9989, a recall of 0.9977, and a precision of 0.9869, indicating that our approach provides higher accuracy than that of the two previous implementations. Possible explanations for the performance deterioration in the two implementations include the following. First, there are natural limitations to how well benign and malicious domains can be identified based on character patterns alone. In the implementation based on [5], the tendency to identify domain names consisting of eight or more letters as malicious was particularly prominent, and the results caused more than 15,000 false positives for benign domains X_0. Next, in the implementation based on [36], uniformly quantifying the importance of words in the domain strings resulted in a large number of misidentified domains. Specifically, the cases of benign domains X_0 and Pizd domains X_4 were significant, exceeding 100,000 and 23,000, respectively. While Pizd misidentifications might be improved by optimizing the threshold level, false positives for benign domains will also increase. The ability of our proposed approach to navigate these challenges can be attributed to the consideration of differences between words constituting benign and malicious domain names and to its focus on words playing central roles in word graphs.

Table 5. Experimental results.

	Accuracy	**Recall**	**Precision**
Anderson et al. [5]	0.9950	0.9977	0.9305
Pereira et al. [36]	0.9599	0.8873	0.6380
Our work	0.9989	0.9977	0.9869

The numbers of misidentified domains are presented in Table 6, where each value is the sum of the five-fold cross-validation results. Our approach achieves an extremely strong identification performance for Banjori X_1 and Sisron X_6, whereas the other datasets, including the benign domain names, result in some misidentification. Notably, Banjori X_1 and Sisron X_6 generate domain names by combining random characters with specific words, and the other types generate domain names by concatenating words contained in their own dictionary. Most of these misidentified domain names can be classified into four categories: (a) Domain names consisting of a single word, (b) domain names consisting of words that appear with extremely low frequency in the training datasets, (c) domain names containing some words common to both benign and malicious domains, and (d) domain names comprising words not included in the dictionary. The proposed approach focuses on the relationships among words in domain names and leverages supervised ML to identify benign and malicious domains. However, the properties of this approach naturally make accurate identification difficult in cases (a) and (b). An investigation into the character strings arising in case (c) revealed numerous occurrences of words such as `domain`, `network`, `host`, and `local`, which are common to both benign and malicious domains. Therefore, it may be possible to improve the performance in this area by excluding words that commonly appear in domain names by following the technique of stop words in natural language processing [49]. The poor performance in case (d) arose from domain names being segmented into meaningless short character strings. Examples of this problem include domain names composed of random character strings, nonalphabetical strings represented alphabetically, or proper nouns. Thus, the use of more comprehensive dictionaries is expected to alleviate this issue.

Table 6. Numbers of misidentified domains in the datasets.

	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7
Anderson et al. [5]	15,631	0	154	0	62	27	0	220
Pereira et al. [36]	105,706	0	285	26	23,286	28	0	36
Our work	2772	0	295	31	29	32	0	82

Despite the above problems, the proposed approach achieves high performance, with an accuracy of 0.9989, a recall of 0.9977, and a precision of 0.9869. These results suggest that dict-DGA malware can be removed from a network using name resolution for malicious domains as triggers.

4.3. Qualitative Comparison

Table 7 presents a qualitative comparison of the proposed approach and nine previously published methods, which are representative examples of blacklist-based detection [18], DPI-based detection [23], behavior-based detection [29–31], ML-based detection [4], DL-based detection [5,34], and estimated dictionary-based detection [36]. In this comparison, we focus on the following five items: (1) DGA malware detection performance, (2) dict-DGA malware detection performance, (3) the ability to detect malware in real time, (4) robustness to encryption, and (5) dependence on network scale. These items are used to compare detection performance and operational limitations, which are the practical issues of each method discussed in Sections 2.2 and 4.2. The first three items correspond to detection performance, while the other two correspond to operational limitations.

Table 7. Qualitative comparison of our work with other well-known detection methods.

	(1) DGA Malware Detection	**(2) Dict-DGA Malware Detection**	**(3) Real-Time Detection**	**(4) Robust to Encryption**	**(5) Network-Scale Independent**
Soldo et al. [18]			✓	✓	✓
Gu et al. [23]			✓		✓
Rahbarinia et al. [29]			✓	✓	✓
Berger et al. [30]				✓	
Wang et al. [31]	✓	✓		✓	
Truong et al. [4]	✓		✓	✓	✓
Anderson et al. [5]	✓		✓	✓	✓
Qiao et al. [34]	✓		✓	✓	✓
Pereira et al. [36]		✓	✓	✓	✓
Our work		✓		✓	✓

As noted in the previous section, the proposed approach has the ability to detect dict-DGA malware with high accuracy. The specific advantages of this approach include the following: In contrast to the BotHunter of Gu et al. [23], the detection performance is not limited by encrypted communication; in contrast to both the DNSMap of Berger et al. [30] and the DBod of Wang et al. [31], the operation does not depend on the network scale. Unfortunately, our approach is useless against typical DGA malware because it specializes in detecting dict-DGA malware and thus must be complemented by additional techniques, such as the methods of Anderson et al. [5] and Qiao et al. [34], for practical operation.

The proposed approach is more computationally expensive than other detection methods based on character strings in domain names [4,5,34,36] because of the two tasks must be performed via brute-force algorithms in Sections 3.3 and 3.4: Identifying optimal candidates to segment domain strings into words and associating vertices with words. As described in

Section 4.1, our approach took 2896 s to process 646,224 queries while the calculation times for [5,36] were 402 s and 72 s, respectively. To reduce the computational cost, this approach restricts attention solely to specific domains for which the name resolution meets both the request of forward lookup and the response of NXDOMAIN messages. Imposing these two conditions substantially limits the number of domains that must be identified. Note that an NXDOMAIN response signifies that a domain is not registered, so no data transmission can arise associated with the name resolution. Consequently, since our approach does not require strict real-time identification, simple optimization should suffice to yield acceptable performance for practical operation.

The importance of encrypted communication in networks has spurred efforts to standardize protocols such as DNS over TLS (DoT) [50], which encapsulates DNS queries and their responses inside TLS communications to prevent eavesdropping and tampering. Since the proposed approach relies solely on the character strings of domain names, it works unaffected by DoT encryption under the condition that the query logs can be retrieved from the RDNSs. Meanwhile, it is powerless when DGA malware directs queries to the external RDNSs, such as Google and Cloudflare Public DNSs. A guideline published by National Security Agency [51] recommends blocking name resolution via such unauthenticated RDNSs because of security and privacy concerns. Consequently, the condition does not compromise the practicality of our approach in typical network operation.

Our approach, similarly to all ML- and DL-based methods, requires the preparation of large datasets to achieve high performance. Several projects have released datasets as part of their results, and several studies have focused on the accumulation of datasets themselves. For example, Kountouras et al. [52] implemented a system called Thales, which creates massive amounts of malicious domain names by distilling multiple freely available sources, while Pearce et al. [53] developed a scalable, accurate, and ethical system, called Iris, which measures global name resolution and uses active manipulation to track the trends of domain names that evolve over time. In addition, Viglianisi et al. [54] referred to SysTaint, which facilitates reverse engineering of malware communications. Therefore, the increasing importance of ML and DL techniques for security applications may be expected to further stimulate activities associated with expanding the datasets on which such methods rely.

The above discussion demonstrates the superiority of our proposed approach for detecting dict-DGA malware. Specifically, our approach allows the elimination of infected computers without restrictions due to encrypted communication or network scale, thereby helping to ensure secure network operations.

5. Conclusions

In this paper, we aimed to identify dynamically generated domains from massive DNS queries to detect the callbacks of dict-DGA malware. The main issue to address was the character-level similarities between dynamically generated domain names and human-generated domain names; modeling based on these similarities is unlikely to be sufficiently accurate to detect dict-DGA malware. Therefore, we proposed an approach for identifying malicious domain names by analyzing their character strings at the word level. This approach was based on the following observations: Each type of dict-DGA malware uses its own dictionaries and algorithms to generate domain names and the word usages of malware-generated domains are distinctly different from those of human-generated domains. The results of the experiments conducted as part of this study showed that our approach could achieve accuracy, recall, and precision as high as 0.9989, 0.9977, and 0.9869, respectively. We also clarified the functional superiority of our approach via qualitative comparisons with published methods. Taken together, these results suggest that malware-infected computers can be identified and removed from networks using DNS queries for detected malicious domains as triggers. Therefore, we conclude that our approach can contribute considerably to improving network security by providing the ability to address various malware encroachments.

The novel characteristics of our strategy ensure three levels of effectiveness. First, our approach enables high performance for detecting dict-DGA malware, even though it lacks the ability to detect typical DGA malware. It may be possible to resolve this issue by combining this approach with other available techniques. Next, the applicability of our approach is not restricted to networks with encrypted communications. In addition, its nature does not depend on the network scale. Finally, the computational cost of our approach can be reduced by restricting attention solely to specific domains for which the name resolution meets both the request of forward lookup and the response of NXDOMAIN messages. Imposing these two conditions substantially reduces the number of domains requiring identification, which means that simply optimizing this approach should suffice to yield satisfactory performance for practical operation.

In the future, we plan to evaluate the identification accuracy and computational time of our proposed approach in relation to DNS queries observed over large-scale networks. We will also continue to discuss the evaluation results.

Author Contributions: Conceptualization and methodology, A.S. and Y.F.; software, A.S.; validation, formal analysis, and investigation, A.S. and Y.F.; resources, G.K. and Y.N.; data curation, A.S.; writing—original draft preparation and writing—review and editing, A.S.; visualization, A.S. and Y.F.; supervision and project administration, G.K. and Y.N.; funding acquisition, A.S. and G.K.; All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Telecommunications Advancement Foundation (TAF). Part of this work was carried out as a Cooperative Research Project of the Research Institute of Electrical Communication (RIEC) of Tohoku University.

Data Availability Statement: DNS queries observed in our institute cannot be shared due to legal, confidentiality, and privacy concerns.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are mainly used in this manuscript:

DGA	Domain Generation Algorithm
Dict-DGA	Dictionary-Based DGA
C&C	Command-and-Control Server
DNS	Domain Name System
ADNS	Authoritative DNS Server
RDNS	Recursive DNS Server
NXDOMAIN	Nonexisting Domain

References

1. Lewis, J.A. Economic Impact of Cybercrime—No Slowing Down, 2018. Available online: https://www.csis.org/analysis/economic-impact-cybercrime (accessed on 1 November 2020).
2. Fu, Y.; Yu, L.; Hambolu, O.; Ozcelik, I.; Husain, B.; Sun, J.; Sapra, K.; Du, D.; Beasley, C.T.; Brooks, R.R. Stealthy Domain Generation Algorithms. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 1430–1443. [CrossRef]
3. Satoh, A.; Nakamura, Y.; Nobayashi, D.; Ikenaga, T. Estimating the Randomness of Domain Names for DGA Bot Callbacks. *IEEE Commun. Lett.* **2018**, *22*, 1378–1381. [CrossRef]
4. Truong, T.; Guang, C. Detecting Domain-Flux Botnet based on DNS Traffic Features in Managed Network. *Secur. Commun. Networks* **2016**, *9*, 2338–2347. [CrossRef]
5. Anderson, H.S.; Woodbridge, J.; Filar, B. DeepDGA: Adversarially-Tuned Domain Generation and Detection. In Proceedings of the ACM Workshop on Artificial Intelligence and Security, Vienna, Austria, 28 October 2016; pp. 13–21.
6. Sood, A.K.; Zeadally, S. A Taxonomy of Domain-Generation Algorithms. *IEEE Secur. Priv.* **2016**, *14*, 46–53. [CrossRef]
7. Zhauniarovich, Y.; Khalil, I.; Yu, T.; Dacier, M. A Survey on Malicious Domains Detection through DNS Data Analysis. *ACM Comput. Surv.* **2018**, *51*, 67. [CrossRef]
8. Chen, J.; Hu, Y.; Liu, J.; Xiao, Y.; Jiang, H. Deep Short Text Classification with Knowledge Powered Attention. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; pp. 6252–6259.
9. Xia, T.; Chen, X. A Discrete Hidden Markov Model for SMS Spam Detection. *Appl. Sci.* **2020**, *10*, 5011. [CrossRef]

10. Baccouche, A.; Ahmed, S.; Sierra-Sosa, D.; Elmaghraby, A. Malicious Text Identification: Deep Learning from Public Comments and Emails. *Information* **2020**, *11*, 312. [CrossRef]
11. Hacker Target Pty. Ltd. Download Top 1 Million Sites. Available online: https://hackertarget.com/top-million-site-list-download/ (accessed on 1 November 2020).
12. Oz, H.; Aris, A.; Levi, A.; Uluagac, A.S. A Survey on Ransomware: Evolution, Taxonomy, and Defense Solutions. *arXiv* **2021**, arXiv:2102.06249.
13. Truong, T.C.; Diep, Q.B.; Zelinka, I. Artificial Intelligence in the Cyber Domain: Offense and Defense. *Symmetry* **2020**, *12*, 410. [CrossRef]
14. Yurekten, O.; Demirci, M. SDN-Based Cyber Defense: A Survey. *Future Gener. Comput. Syst.* **2021**, *115*, 126–149. [CrossRef]
15. Kim, D. Potential Risk Analysis Method for Malware Distribution Networks. *IEEE Access* **2019**, *7*, 185157–185167. [CrossRef]
16. Cai, Y.; Yee, G.; Gu, Y.X.; Lung, C. Threats to Online Advertising and Countermeasures: A Technical Survey. *ACM Digit. Threat. Res. Pract.* **2020**, *1*, 1–27. [CrossRef]
17. Andriesse, D.; Rossow, C.; Stone-Gross, B.; Plohmann, D.; Bos, H. Highly Resilient Peer-to-Peer Botnets Are Here: An Analysis of GameOver Zeus. In Proceedings of the International Conference on Malicious and Unwanted Software, Fajardo, PR, USA, 22–24 October 2013; pp. 116–123.
18. Soldo, F.; Le, A.; Markopoulou, A. Blacklisting Recommendation System: Using Spatio-Temporal Patterns to Predict Future Attacks. *IEEE J. Sel. Areas Commun.* **2011**, *29*, 1423–1437. [CrossRef]
19. Freudiger, J.; Cristofaro, E.; Brito, A. Controlled Data Sharing for Collaborative Predictive Blacklisting. In Proceedings of International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Milan, Italy, 9–10 July 2015; pp. 327–349.
20. Špaček, S.; Laštovička, M.; Horák, M.; Plesník, T. Current Issues of Malicious Domains Blocking. In Proceedings of the IFIP/IEEE International Symposium on Integrated Network and Service Management, Arlington, VA, USA, 8–12 April 2019; pp. 551–556.
21. Špaček, S.; Rusňák, V.; Dombajová, A. DNS Firewall Data Visualization. In Proceedings of the IFIP/IEEE International Symposium on Integrated Network and Service Management, Arlington, VA, USA, 8–12 April 2019; pp. 743–744.
22. Vixie, P.; Schryver, V. *DNS Response Policy Zones (RPZ)*; IETF Internet Draft: Draft-vixie-dnsop-dns-rpz-00; IETF: Fremont, CA, USA, 2018.
23. Gu, G.; Porras, P.; Yegneswaran, V.; Fong, M.; Lee, W. BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation. In Proceedings of the USENIX Conference on Security Symposium, Anaheim, CA, USA, 9–11 August 2007; pp. 167–182.
24. Parvat, T.J.; Chandra, P. Performance Improvement of Deep Packet Inspection for Intrusion Detection. In Proceedings of the IEEE Global Conference on Wireless Computing & Networking, Lonavala, India, 22–24 December 2014; pp. 224–228.
25. Su, J.; Chen, S.; Han, B.; Chengcheng, X.; Wang, X. A 60Gbps DPI Prototype based on Memory-Centric FPGA. In Proceedings of the ACM SIGCOMM Conference, Florianopolis, Brazil, 22–26 August 2016; pp. 627–628.
26. Cisco Systems Inc. Cisco Annual Cybersecurity Report 2018. Available online: https://www.cisco.com/c/en/us/products/security/cybersecurity-reports.html (accessed on 1 November 2020).
27. Poh, G.S.; Divakaran, D.M.; Lim, H.W.; Ning, J.; Desai, A. A Survey of Privacy-Preserving Techniques for Encrypted Traffic Inspection over Network Middleboxes. *arXiv* **2021**, arXiv:2101.04338.
28. Satoh, A.; Nakamura, Y.; Fukuda, Y.; Sasai, K.; Kitagata, G. A Cause-Based Classification Approach for Malicious DNS Queries Detected Through Blacklists. *IEEE Access* **2019**, *7*, 142991–143001. [CrossRef]
29. Rahbarinia, B.; Perdisci, R.; Antonakakis, M. Efficient and Accurate Behavior-Based Tracking of Malware-Control Domains in Large ISP Networks. *ACM Trans. Priv. Secur.* **2016**, *19*, 1–31. [CrossRef]
30. Berger, A.; D'Alconzo, A.; Gansterer, W.; Pescapè, A. Mining Agile DNS Traffic Using Graph Analysis for Cybercrime Detection. *Comput. Netw.* **2016**, *100*, 28–44. [CrossRef]
31. Wang, T.; Lin, H.; Cheng, W.; Chen, C. DBod: Clustering and Detecting DGA-based Botnets using DNS Traffic Analysis. *Comput. Secur.* **2017**, *64*, 1–15. [CrossRef]
32. Plohmann, D.; Yakdan, K.; Klatt, M.; Bader, J.; Gerhards-Padilla, E. A Comprehensive Measurement Study of Domain Generating Malware. In Proceedings of the USENIX Conference on Security Symposium, Austin, TX, USA, 10–12 August 2016; pp. 263–278.
33. Zago, M.; Gil Pérez, M.; Martínez Pérez, G. UMUDGA: A Dataset for Profiling DGA-based Botnet. *Comput. Secur.* **2020**, *92*, 101719. [CrossRef]
34. Qiao, Y.; Zhang, B.; Zhang, W.; Sangaiah, A.K.; Wu, H. DGA Domain Name Classification Method Based on Long Short-Term Memory with Attention Mechanism. *Appl. Sci.* **2019**, *9*, 4205. [CrossRef]
35. Vinayakumar, R.; Soman, K.; Poornachandran, P.; Kumar, S. Evaluating Deep Learning Approaches to Characterize and Classify the DGAs at Scale. *J. Intell. Fuzzy Syst.* **2018**, *34*, 1265–1276. [CrossRef]
36. Pereira, M.; Coleman, S.; Yu, B.; DeCock, M.; Nascimento, A. Dictionary Extraction and Detection of Algorithmically Generated Domain Names in Passive DNS Traffic. In Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses, Heraklion, Greece, 10–12 September 2018; pp. 295–314.
37. Koren, A. Ursnif Malware: Deep Technical Dive. Available online: https://arielkoren.com/blog/2016/11/01/ursnif-malware-deep-technical-dive (accessed on 1 November 2020).

38. Skuratovich, S. Matsnu: A Deep Dive. Available online: https://blog.checkpoint.com/2015/07/02/matsnu-a-new-malware-discovery/ (accessed on 1 November 2020).
39. Sahoo, D.; Liu, C.; Hoi, S. Malicious URL Detection using Machine Learning: A Survey. *arXiv* **2019**, arXiv:1701.07179v3.
40. Levine, J. *DNS Blacklists and Whitelists*; IETF Request for Comments: 5782; IETF: Fremont, CA, USA, 2010.
41. Mockapetris, P. *Domain Names—Implementation and Specification*; IETF Request for Comments: 1035; IETF: Fremont, CA, USA, 1987.
42. Costello, A. *Punycode: A Bootstring Encoding of Unicode for Internationalized Domain Names in Applications (IDNA)*; IETF Request for Comments: 3492; IETF: Fremont, CA, USA, 2003.
43. Liu, B.; Lu, C.; Li, Z.; Liu, Y.; Duan, H.; Hao, S.; Zhang, Z. A Reexamination of Internationalized Domain Names: The Good, the Bad and the Ugly. In Proceedings of the Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Luxembourg, 25–28 June 2018; pp. 654–665.
44. Chiba, D.; Hasegawa, A.; Koide, T.; Sawabe, Y.; Goto, S.; Akiyama, M. DomainScouter: Analyzing the Risks of Deceptive Internationalized Domain Names. *IEICE Trans. Inf. Syst.* **2020**, *E103-D*, 1493–1511. [CrossRef]
45. Fraunhofer FKIE. DGArchive. Available online: https://dgarchive.caad.fkie.fraunhofer.de (accessed on 1 November 2020).
46. Bader, J. Some Results of My DGA Reversing Efforts. Available online: https://github.com/baderj/domain_generation_algorithms (accessed on 1 November 2020).
47. Atkinson, K. GNU Aspell. Available online: http://aspell.net (accessed on 1 November 2020).
48. Norvig, P. Natural Language Corpus Data: Beautiful Data. Available online: http://norvig.com/ngrams/ (accessed on 1 November 2020).
49. Nothman, J.; Qin, H.; Yurchak, R. Stop Word Lists in Free Open-source Software Packages. In Proceedings of the Workshop for NLP Open Source Software, Melbourne, Australia, 15–20 July 2018; pp. 7–12.
50. Hu, Z.; Zhu, L.; Heidemann, J.; Mankin, A.; Wessels, D.; Hoffman, P. *Specification for DNS over Transport Layer Security (TLS)*; IETF Request for Comments: 7858; IETF: Fremont, CA, USA, 2016.
51. National Security Agency. Adopting Encrypted DNS in Enterprise Environments, 2021. Available online: https://media.defense.gov/2021/Jan/{\protect\penalty\z@}14/2002564889/-1/-1/0/csi_adopting_encrypted_dns_u_oo_102904_21.pdf (accessed on 20 April 2021).
52. Kountouras, A.; Kintis, P.; Lever, C.; Chen, Y.; Nadji, Y.; Dagon, D.; Antonakakis, M.; Joffe, R. Enabling Network Security Through Active DNS Datasets. In Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses, Paris, France, 19–21 September 2016; pp. 188–208.
53. Pearce, P.; Jones, B.; Li, F.; Ensafi, R.; Feamster, N.; Weaver, N.; Paxson, V. Global Measurement of DNS Manipulation. In Proceedings of the USENIX Security Symposium, Vancouver, BC, Canada, 16–18 August 2017; pp. 307–323.
54. Viglianisi, G.; Carminati, M.; Polino, M.; Continella, A.; Zanero, S. SysTaint: Assisting Reversing of Malicious Network Communications. In Proceedings of the Software Security, Protection, and Reverse Engineering Workshop, San Juan, PR, USA, 3–4 December 2018; pp. 1–12.