Received February 26, 2020, accepted March 30, 2020, date of publication April 3, 2020, date of current version April 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2985367

REMaDD: Resource-Efficient Malicious Domains Detector in Large-Scale Networks

OFIR ERETS KDOSHA[1], **GILAD ROSENTHAL**[1], **KOBI COHEN**[1], **ALON FREUND**[2], **AVISHAY BARTIK**[2], **AND AVIV RON**[2]

[1]School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beer Sheva 8410501, Israel
[2]IBM Cyber Security Center of Excellence, Beer Sheva 8489325, Israel

Corresponding author: Kobi Cohen (yakovsec@bgu.ac.il)

This work was supported in part by the IBM Cyber Security Center of Excellence, Gav-Yam Negev, and in part by the U.S.-Israel Binational Science Foundation (BSF) under Grant 2017723.

ABSTRACT Detecting malicious activities in cyber systems is a major challenge of cybersecurity service providers. Due to the large amount of network traffic, it is often likened to finding a needle in a haystack. Domain name system (DNS) is one of the fundamental protocols of the internet, and therefore it can give a broad view of those malicious activities, which abuse it and leave fingerprints as part of their attack vector. In this collaborative research between Ben-Gurion University, and IBM, a significant performance improvement was achieved in detecting malicious domains as compared to the state-of-the-art software solutions. Specifically, we establish a novel algorithm to detect malicious domains in large-scale DNS traffic, named Resource-Efficient Malicious Domain Detector (REMaDD), with the following desired properties. First, the algorithm does not require prior knowledge on historical malicious activities in its real-time operations. Second, the development used real live streaming data from The Inter-University Computation Center (IUCC), and operated on real-time IBM system. The algorithm is highly computational efficient and satisfies real-time requirements in terms of running time and computational complexity. REMaDD demonstrated strong performance in terms of both detection accuracy and computational efficiency as compared to existing algorithms. Specifically, experimental results on IBM production environment demonstrated that REMaDD achieved 89.4% Precision score, and 82.9% Recall score. By contrast, the DomainObserver, and LSTM.MI algorithms achieved only 76.7%, 67.2% Precision score, and 81.7%, 75.3% Recall score, respectively.

INDEX TERMS Cyber security, domain name system (DNS), detection algorithms, real-time algorithms.

I. INTRODUCTION
Nowadays, modern society relies on the internet in many aspects of public services as well as private life. This reliance increases the exposure to cyber threats and raises many security challenges with respect to user privacy, integrity, and availability. These challenges have triggered the need to finding cyber security solutions to a variety of potential cyber attacks to ensure safe internet environment.

The DNS protocol is one of the fundamental protocols of the internet, as it allows users to conveniently identify the IP of internet resources, such as computers, networks, and services by their names [1]. The DNS is often abused by cyber criminals to launch different kinds of malicious activities. Since purchasing a domain name is very easy and accessible, cyber criminals often buy and operate them for malicious purposes, such as phishing, botnets, ransomware, etc. [2], [3]. This common malicious usage of the DNS has triggered an extensive research for detecting malicious domains. This is a rather challenging task, since new malicious domains emerge every day and the domain name space is ‘‘almost infinity’’. Furthermore, since cybercrimes are very profitable for the criminals, the crime industry grows increasingly sophisticated. As a result, techniques of operating malicious domains are constantly evolving and changing, making them harder to detect [4].

Detecting malicious activities in large-scale networks is even more challenging, as one cannot check every single domain name query. Moreover, saving statistical data on every queried domain is impractical, since the memory requirement can easily grow to an enormous size. Computing complicated models using machine learning algorithms on

The associate editor coordinating the review of this manuscript and approving it for publication was Yu-Chi Chen.

such large databases can be very slow, and might lead to a delayed detection that will not allow preventing the attack effectively. For these reasons, it is important to judiciously exploit patterns in the DNS traffic, which allows labeling certain domain names as malicious quickly and reliably. Furthermore, using passive traffic measurements (i.e., measurements that were observed from the network traffic passively, without creating or modifying any traffic on the network) is advantageous in detection algorithms, since the network remains in its true pattern state, and overloading the network is avoided.

The objective of this research is to develop an algorithm for detecting malicious domains in large-scale networks based on passive traffic measurements. The algorithm must have the ability to operate effectively in real-time environments, which contain a massive amount of DNS traffic data. Therefore, it must be light weight in terms of computational resources and memory usage, and still be able to achieve the state-of-the-art performance in terms of detection accuracy.

A. MAIN RESULTS

Most of the existing studies on malicious domains detection have tackled the problem by using offline machine learning algorithms or signature based techniques (see [5] and related work in Section I-B). Signature-based techniques are able to detect only known bots through signature matching and are based on black lists such as DNS-based blacklist (DNSBL) [6]. Those methods have been successfully applied in security software for detecting attacks with known signatures. However, they tend to fail when it comes to detecting new malicious domains in large-scale networks, since it is generally impossible to maintain an updated list of malicious domains when new ones emerge every day. Moreover, most of those techniques were developed on closed datasets, and sometimes even synthetic datasets. Therefore, their performance on real data in real-time systems is unclear.

In [7], the DomainObserver algorithm was developed, and achieved the state-of-the-art performance in time series malicious domain detection. DomainObserver shares similarities with our approach in the sense that it applies passive traffic measurements and time series data mining techniques to detect malicious domains. The authors [7] gathered three types of time series data for each domain: access, users, and entropy, which are constructed from aggregated traffic measurements for a certain time window. A K-Nearest Neighbor (KNN) classifier was applied using Dynamic Time Warping (DTW) as the distance metric on each domain's series. The dataset used in [7] was a closed dataset of verified 1296 malicious and 373 benign domains activity for 94 hours, gathered from real backbone network traffic measurements. According to the verification results in [7], each one of the three types of time series gave approximately the same classification results.

The DomainObserver algorithm cannot operate effectively, however, in a large-scale system environment considered in this research due to several reasons. First, the users and entropy time series require the client's IP address which is unavailable in many environments due to privacy and topology issues. Second, a 94 hours long measurement renders the algorithm unfit for many real-time scenarios since the defending enterprise wishes to detect threats much earlier to avoid damage to the system. A more acceptable measurement time in real-time domain detection is 12 hours, which is the delay constraint considered in this research. Third, the amount of DNS queries passing through a typical enterprise system makes it extremely difficult to apply the DomainObserver due to scaling issues on large datasets. In this paper we overcome these issues.

Below, we summarize our main contributions:

1) REAL DATA ANALYSIS

We observe and study the access patterns of malicious and benign domains, based on real live streaming passive traffic measurements. Traffic spikes were observed in various forms of cyber-attacks. Our observations confirm the conjecture about the correlation between malicious domains and spikes in their access patterns. Furthermore, we have observed that a similar correlation applies to benign domains for Internet of Things (IoT) product companies, when synchronizing or sending update requests periodically. Fig. 4 in Section IV illustrates the different access pattern of each domain group. These important insights on access patterns were used in the algorithm development.

2) ALGORITHM DEVELOPMENT

We establish a novel computational efficient real-time algorithm, named Resource-Efficient Malicious Domain Detector (REMaDD), for detecting malicious domains in large-scale networks. The algorithm works as follows. First, the algorithm implements a low-complexity filtering of the live streaming data by judiciously detecting spikes online in the DNS traffic measurements. Based on our data analysis, these traffic spikes are used as access patterns in the filtering process. This detection phase declares suspects quickly and significantly reduces the number of tested domains for the next deeper inspection phase. It allows the algorithm to run a deeper inspection phase with very low computational consumption, run time, and memory usage. In the next phase, we develop a real-time deep inspection domain classifier to label the domains as malicious or benign. We develop real-time Random Forest classification in this phase.

It is worth noting that real-time deployment of the DomainObserver algorithm remained open in [7] since it is much more demanding in terms of computational resources. This issue has been pointed out by the authors as a future research direction (see the Future Work section in [7]). In this paper, the development of REMaDD solves this issue. REMaDD is highly efficient, and can be trained and deployed in real-world enterprise scenarios as tested at IBM cyber security lab in this research on the IUCC network.

3) ACHIEVING STRONG PERFORMANCE IN A REAL-TIME SYSTEM

It is well known that enterprises do not eager to share their real data and real-time cyber security performance due to legal risks, such as violating privacy, or due to economic incentives, such as sharing information that will benefit their competitors. Therefore, analyzing and validating cyber security algorithms in real-time systems using up-to-date real data is a main limitation of the academic cyber security research. This collaborative research between Ben-Gurion University, and IBM cyber security lab enabled us to analyze and validate the algorithm performance in a real-time system using up-to-date real live streaming data. To evaluate the algorithm performance, we deployed it in a real working environment (see Section II for details) and recorded the results for two different periods of two weeks. Note that labeling domain names as malicious or benign for performance evaluation is a difficult task, since new and esoteric domain names appear frequently. Therefore, researchers often rely on available datasets, which might be irrelevant in a real-world environment. By contrast, in order to evaluate the most up-to-date results in this research, two cyber security researchers have invested 30 full-time days to label up-to-date real-data at IBM cyber security lab (see more details in Section IV-A).

The real-time detection performance and resource consumption achieved by REMaDD algorithm were satisfying and highly compatible in both tests. Specifically, the detection performance obtained by REMaDD algorithm outperformed the state-of-the-art performance of time series detection obtained by the DomainObserver algorithm. Furthermore, REMaDD saves significant computational resources as compared to the DomainObserver algorithm. The comparison results are presented in details in Section IV.

B. RELATED WORK

As discussed in Section I-A, most of the malicious domain detection techniques are based either on machine learning algorithms [7]–[22] or signature-based algorithms [6], [23]–[25]. Signature-based algorithms rely on cyber security expert insights obtained by measurement studies, which explored the behavior of known malicious domains [26]. For instance, in [24], the authors introduced the BotDet system which is a real-time system for detecting traffic related to botnets command and control servers. The system was composed by four different detection modules and a correlation framework that correlates between the module results. Three of the modules were based on black lists and the last was based on DNS query failures.

The family of machine learning based algorithms can be divided into two groups. The first group is host-based algorithms, where monitoring and analyzing processes are made locally at each individual computer [8], [27]. The second group is network-based algorithms, which monitor and analyze the network traffic to identify the existence of malicious domains.

In past and recent years, various network-based techniques have been proposed for malicious domains detection. In [9], the authors developed an anomaly-based botnet detection mechanism by monitoring network clients' activities. The algorithm detects a similar activity of a group of clients querying for the same domain in subsequent time periods, or for several different domains with approximately the same number of clients in the same period. In [10], [11], the authors introduced a system of DNS analysis that extracts 15 features from passive DNS traffic measurements. This allows to characterize different domain properties, and detect malicious domains by decision tree approach. In [13], the authors presented a four-stage filtering system to detect malicious domains. The filters are based on domain name group activity, the correlation between DNS response success, and failure and similarity between a pair of domain names. In [14], the authors introduced the BotGAD detector which is based on monitoring group behaviors that appear in the DNS traffic. This detector extracts features from the network traffic measurements to distinguish between malicious and benign DNS queries that might be part of botnet traffic if it appears as a group of hosts showing the same behavior. In [7], the authors applied passive traffic measurements and time series data mining to detect malicious domains, as discussed in Section I-A.

Another family of machine learning-based algorithms uses deep neural networks, e.g., LSTMs and RNNs, for character-based text classification (i.e., NLP). These algorithms classify a domain solely by its name. This method was shown to be highly efficient for domains which were generated by Domain Generation Algorithm (DGA) [19], [20], [22] (i.e., type-specific methods). In this paper, we tested the LSTM.MI algorithm suggested in [20] as a representative algorithm of this method.

In Table 1 we present a qualitative comparison between REMaDD and other related algorithms. In this paper, we focus on detecting unlimited types (or type-free) of malicious domains (where the term unlimited types of malicious domains is borrowed from [7]), as opposed to type-specific methods. Therefore, comparing REMaDD and DomainObserver algorithms that share this property is of a particular interest.

In this paper we focus on a real-time malicious domain detection in large-scale networks. Pure machine learning

TABLE 1. Qualitative algorithm comparison.

Algorithm	Data collection time (hours)	Feature engineering	Scalable	Type of malicious
REMaDD	12	Yes	Yes	Unlimited
DomainObserver [7]	94	No	No	Unlimited
BotGAD [14]	1	Yes	No	Limited[1]
LSTM.MI [20]	0	No	Yes	DGA
DNSBL [6]	0	No	Yes	Known

[1] Incapable of detecting fast-flux or DGA based botnets [28]

algorithms are not compatible in this setting. Specifically, they require scanning all the domains queried by the network and performing feature extraction on each of them, which is too demanding in terms of complexity and time consumption. Thus, those algorithms are not scalable for large networks. NLP algorithms are not compatible as well, since they are based on the domain name exclusively. They perform well in terms of detecting domains which were generated by DGA, but perform poorly in terms of detecting other types of malicious domains. Signature-based algorithms, on the other hand, are efficient and scalable. However, they tend to fail when new malicious domains emerge. As opposed to the techniques mentioned above, we introduce a novel, light weight and computational efficient real-time algorithm for detecting malicious domains which can be implemented in large-scale networks under real-time constraints, as discussed in Section I-A.

II. SYSTEM MODEL AND PROBLEM STATEMENT

In this section we present the DNS model, the packet analysis of the live streaming dataset, and the research objectives.

A. THE DNS MODEL

Consider an Internet Service Provider (ISP)'s DNS recursive resolvers architecture, as illustrated in Fig. 1. Each group of clients is connected to a local DNS recursive resolver which provides DNS services for that group. The DNS recursive resolvers are connected to the World Wide Web (WWW), so they could query authoritative DNS servers about the DNS information for their clients. Typically, when a DNS client needs to find the IP address of a host or service, known by its Fully Qualified Domain Name (FQDN), it queries its DNS recursive resolver for the IP Address. The recursive resolver first looks for the IP address in its cache. If it does not exist, it starts making a hierarchical recursive resolution process, which starts with the root servers and ends at an authoritative name server. The DNS system is designed to be hierarchical in the form of a tree data structure. The root node contains the addresses of all Top Level Domain (TLD) name servers, and these servers contain the addresses of Second Level Domain (SLD) name servers, and so on. Therefore, the search starts at the root node and continues with the next tree level, until an authoritative answer is found which yields the answer to the requested query. Additionally, in a case the FQDN is invalid or non existent in the tree, the recursive resolver returns this information to the client.

B. PACKET ANALYSIS OF THE LIVE STREAMING DATA

The data for this work is provided by the Inter-University Computation Center (IUCC), the ISP of the academic institutes in Israel. The data is in the form of a real-time passive DNS inspection. To gather the DNS records, a packet capture mechanism (sniffer) was placed above the DNS recursive resolver layer, as can be seen in Fig. 1. This sniffer gathers every DNS query and its response, and streams those records

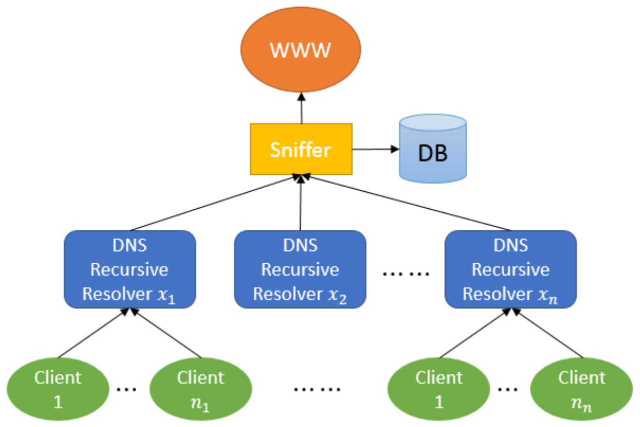

WWW
Sniffer
DB
DNS Recursive Resolver x_1
DNS Recursive Resolver x_2
DNS Recursive Resolver x_n
Client 1
Client n_1
Client 1
Client n_n

FIGURE 1. The IUCC DNS servers architecture.

in real-time to a database. Every record contains all the data of the DNS query and its response excluding the client IP address, due to privacy restrictions. The database used in this research contains two data collections, one contains the raw data and the other contains an hourly aggregation of query counts per SLD. The hourly aggregation optimizes the algorithm's queries for data to make sure that the method can cope with a real-world scenario. In addition, The DNS traffic, which is captured and recorded in the IUCC network has a massive volume. For example, on average, at each hour, the system adds 9.8 million new records of $52,000$ unique domains, and every day, $320,000$ unique domains DNS traffic pass through the system. Therefore, each collection has a retention policy (i.e. each record has a predefined life expectancy). The raw collection retains its records for three days, while the SLD collection retains its records for two months.

Due to the massive volume of evolving real data, and the requirement of detecting cyber threats in real-time and with limited computational resources, the training process of the detection algorithm was implemented using the real-time streaming data. As a result, the entire development of the algorithm was made using up-to-date data from the database mentioned above. The use of real live streaming data at each development stage guarantees that the algorithm meets its resource restrictions, and performs strongly (as will be demonstrated later) in a real-time working environment.

C. THE OBJECTIVE

We next define the commonly used detection measures in the cyber security literature. Let TP, FN, FP and TN denote the number of True Positive (i.e., a malicious domain is classified as malicious), False Negative (i.e., a malicious domain is classified as benign), False Positive (i.e., a benign domain is classified as malicious) and True Negative (i.e., a benign domain is classified as benign) binary classification results, respectively. Let

$$P \triangleq \frac{TP}{TP + FP} \tag{1}$$

denote the Precision score, and let

$$R \triangleq \frac{TP}{TP + FN} \tag{2}$$

denote the Recall score.

Note that in cyber security systems, domains which are declared as malicious are blocked, or assigned to a security operations center (SOC) analyst to investigate and act for each positive case. As a result, high precision implies that among all domains which are declared as malicious, there is a small number of benign domains. In order to minimize false positives in an unbalanced data which is mostly benign, it is often desired to maximize the precision under a target constraint R_t on the recall.

The objective is thus to develop an algorithm that maximizes P under the constraint that $R \geq R_t$.

In this research, we set the target recall rate to $R_t = 0.8$, which is a typical recall constraint value in many cyber security services. This means that the false negative rate is smaller than one fourth the true positive rate. This false negative rate is considered to be small, as dictated by IBM security service requirements. Nevertheless, the algorithm is general and applies to general values of R_t.

In addition, the algorithm is destined to run on IBM production environment in real-time. Therefore, it must fulfill the following design constraints:

Constraint C1: The computational complexity of the algorithm must be of order $O(N)$, where N is the number of domains.

Constraint C2: The memory usage of the algorithm must be less than 500MB.

Constraint C3: The time complexity of the algorithm for post-processing the collected data samples must be less than one hour.

III. THE REMaDD ALGORITHM

To accomplish the objective defined in Section II-C, the algorithm must be light weight. To achieve real-time performance, we divide the detector into four main components, as described in Fig. 2. First, a simple white/black list filter is applied for all domains as an easy-to-implement, manually configurable step that is meant to reduce the computational effort by blacklisting detected domains. Then, a crude classification made using a very fast peak detector which is described in Section III-A. Afterwards, a simple existence check is made, to check if the domain exists and resolved to an IP address. This is another easy-to-implement step that filters nonexistent domains which are highly correlated with malicious activities. Additionally, blocking these domains does not affect users. The final step is a real-time deeper inspection via Random Forest Classification (RFC) as described in Section III-B. The first three steps assure that the deeper inspection step (including the feature extraction procedure) is performed over a small number of suspicious domains.

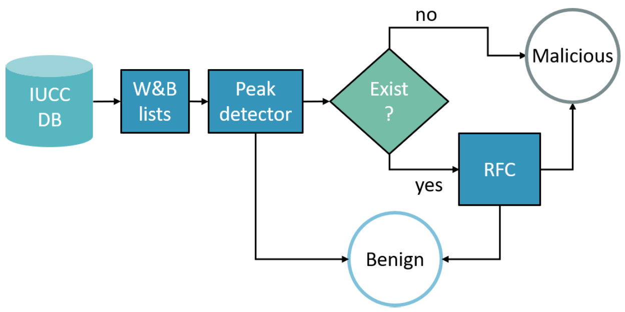


FIGURE 2. The architecture of the REMaDD algorithm.

A. PEAK DETECTOR

The purpose of the peak detector is to detect domains with a sudden abnormal raise in their access pattern and declare them as suspicious domains (which will be passed through deeper inspection in the second phase). The detector is designed to favor false-positive errors over false-negatives, to avoid missing malicious domains. The next phase will perform deeper inspection for a finer classification.

The detector operation consists of two steps which are described formally in (3), (4). The first step of the peak detector is gathering information from the SLD collection and constructing a time series with $N + 1$ data points per each SLD. Each data point is an aggregated sum of the domain's queries over a constant period of time (i.e. the time window). The second step is the detection of the sudden rise in the traffic. This is implemented by computing the weighted average of the first N data points, applying the quantization function described in (5) over the computed weighted average and the last data point, and subtracting between those values. If the difference is bigger than the defined threshold, the domain is considered as suspicious. Mathematically, for each specific second level domain name, denoted by d_j, the peak detector operation is given below:

$$\mathrm{wa}_{d_j}^t = \frac{\sum_{i=1}^{N} a_i w_{d_j}^{t-i \cdot t_s}}{\sum_{i=1}^{N} a_i}, \tag{3}$$

$$\mathrm{dif}_{d_j}^t = Q(w_{d_j}^t) - Q(\mathrm{wa}_{d_j}^t), \tag{4}$$

where t_s is the aggregation period, t denotes the sample time stamp, $w_{d_j}^t$ is the access counts aggregated for a time window t_s, i.e. from time t to $t + t_s$, N is the number of past aggregated time windows (which was set to 5 in our experiments), $a_i, i = 1, \ldots, N$ are the average weights, where higher weights are assigned to recent measurements. The term $\mathrm{wa}_{d_j}^t$ denotes the weighted average of past aggregated time windows for time t. The term $Q(y)$ denotes the quantization function of y as described in Section III-A1, and the term $\mathrm{dif}_{d_j}^t$ denotes the difference between the quantization levels of $w_{d_j}^t$ and $\mathrm{wa}_{d_j}^t$ for domain d_j.

1) THE QUANTIZATION FUNCTION

The quantization function of the DNS traffic count is used to overcome scaling issues due to the high variance in trafic volumes, as well as traffic changes among domains, which were observed while analyzing the data.

On the one hand, since the algorithm tests domains with high variance in their traffic volumes, it should judiciously scale the traffic changes when computing (4). On the other hand, it is likely that traffic changes of p percents in domains with high volume are more significant and might indicate of malicious activity than changes of p percents in domains with low volume (which might occur due to sporadic queries by benign domains for instance).

To solve these issues, we generated an Adaptive-Width quantization method. The boundary values for the quantization levels are given by:

$$q_i = q_{i-1} + d * m^{\lfloor \frac{i}{x} \rfloor}, \tag{5}$$

where d denotes the linear difference between levels, m is a difference multiplier, x is the multiplier cycle, $\lfloor \cdot \rfloor$ is the known floor function, and q_i denotes the quantization level boundary of the ith level. Then, the quantization function in (4) is set to $Q(y) = i$, if $y \in [q_i, q_{i+1})$, where $q_0 = 0$, and the last boundary value equals infinity.

B. SECOND DETECTION PHASE VIA RANDOM FOREST CLASSIFICATION

The second detection phase runs deeper inspection. The goal of this phase is to distinguish between the suspects raised by the pre-filtering and peak detector steps, while meeting the real-time system requirements described in Section II-C.

For this step, several classification methods were trained and tested using features extracted from the DNS traffic measurements and from Whois registration data. The classification method that obtained the best results is the Random Forest method [29]. The Random Forest classification comprised of an ensemble of decision trees, where each tree is constructed by applying algorithm A on the training set S with an additional random vector θ, where θ is sampled i.i.d from some distribution. A prediction of the algorithm is chosen by a majority vote over the prediction of each independent decision tree. The main advantage of developing real-time Random Forest in our setting relies on the randomness it provides. In addition to constructing each tree using different bootstrap sample of the data, by using Random Forest, each node is split using the best predictor among a subset of randomly chosen predictors at the node. This randomness decreases over-fitting and therefore, improving the accuracy of the test set.

The data split of Random Forest can be done using different criteria such as information gain or Gini impurity. In this paper Gini impurity criterion was applied. Gini Impurity is the probability of incorrectly classifying a randomly chosen element in the dataset if it is randomly labeled according to the class distribution in the dataset. To compute Gini impurity for a set of items with J classes, let $i \in \{1, 2, \ldots, J\}$, and p_i be the fraction of items labeled with class i in the set. The Gini Impurity is given by:

$$\begin{aligned} I_G(p) &= \sum_{i=1}^{J} p_i \sum_{k \neq i} p_k = \sum_{i=1}^{J} p_i(1 - p_i) \\ &= \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} p_i^2 = 1 - \sum_{i=1}^{J} p_i^2. \end{aligned} \tag{6}$$

When training a decision tree, the desired split is chosen by maximizing the Gini Gain, which is calculated by subtracting the weighted impurities of the branches from the original impurity.

Developing real-time Random Forest in REMaDD is highly computational-efficient. The output is computed by decision trees, which has low complexity implementations even when handling non-linear dependencies in data. To support our observation, we tested empirically different classification methods, such as SVM, naive-Bayes, KNN, and Adaboost. We indeed found that Random Forest obtained the best performance among all other tested algorithms.

The main features used in the deeper inspection phase are listed below:

- Domain age in days.
- Domain expiration date in days (i.e., days left).
- Days passed from the last update of the domain record.
- Number of FQDN's queried in the last time window.
- Rate of successful A type queries in the last time window.
- Rate of successful AAAA type queries in the last time window.
- Dynamic Time Warping (DTW) distance between the domain time series and the corresponding time series of the previous day.

We started by selecting relevant features judiciously based on extensive experimental analysis that we have made, as well as the system constraints on the feature extraction procedure. Then, we ran the well-known scikit-learn feature importance (i.e., the *feature_importance_* function in ensemble) to obtain an importance score for each feature, and chose high-ranked features. The chosen features are easy to extract from the raw data collection and the WhoIs service, which meets the system constraint on the execution time.

C. PSEUDO CODE OF THE REMaDD ALGORITHM

The pseudo code below describes the algorithm operation at time t. Each domain d_j which is queried during the current interval ($w_{d_j}^t > 0$) is inspected. The domain is checked using the white/black list filter. The time series $\mathrm{TS}_{d_j}^t$ is then constructed for the domains that were not found in any list. A weighted average $\mathrm{wa}_{d_j}^t$ is then calculated according to (3). The difference between the quantization levels of the current sample and the weighted average is computed according to (4) and compared to a threshold. Domains that pass the threshold are checked for existence. Valid domains go through the classifier, whereas invalid domains are labeled as malicious. The classifier first filters the data from the Whois service Who_{d_j}, and the DNS raw data $R^t_{d_j}$. Afterward, a feature extraction process is performed, resulting in $F^t_{d_j}$ which are fed into the RFC to label the domain.

Algorithm 1 REMaDD Algorithm

```
 1)  for $d_j$ where $w^t_{d_j} > 0$ do
 2)    if not $d_j$ in classified_list do
 3)      $\text{TS}^t_{d_j} = \textbf{get}\ ([w^{t-ts}_{d_j}, \ldots, w^{t-N\cdot ts}_{d_j}])$
 4)      $\text{wa}^t_{d_j} = \text{Weighted\_Average}\ (TS_{d_j}, a)$
 5)      if $Q(w^t_{d_j}) - Q(wa^t_{d_j}) \geqslant$ threshold do
 6)        if $d_j$ is exists do
 7)          $\text{Who}_{d_j} = \text{Whois\_Registration\_data}(d_j)$
 8)          $R^t_{d_j} = \text{DNS\_Raw\_data}(d_j, t)$
 9)          $F^t_{d_j} = \text{Extract\_Features}\ (R_{d_j}, Who_{d_j})$
10)          $S_{d_j} = \text{Random\_Forest\_Classifier}\ (F_{d_j})$
11)        else
12)          $S_{d_j} = 1$
13)        end if
14)      else
15)        $S_{d_j} = 0$
16)      end if
17)    end if
18)    append $(S_{d_j}, d_j, t)$ to classified_list
19)  end for
```

Finally, we point out that malwares that use command and control communications, typically desire to operate over a long period of time. During this period of time, DNS traffic analysis can be done by REMaDD to detect the threat. Clearly, the faster the detection algorithm detects the threat is better. The experimental results in the next section demonstrate that REMaDD achieves strong performance in this respect as compared to existing algorithms.

IV. EVALUATION AND EXPERIMENTAL RESULTS

We start by describing the dataset labeling method used for evaluation. Then, we present the algorithm settings, and the experiments conducted on the real live streaming data. We compared the REMaDD performance with the Domain-Observer algorithm, which was shown to achieve the state-of-the-art performance in time series malicious domain detection [7].

A. DOMAIN LABELING

We started by gathering labeled data for training and testing. Labeling domain name as malicious or benign might depend on a subjective perspective and can change in time, which makes the labeling phase a tricky mission when it comes to malicious domain detection in cyber security research. Furthermore, new and esoteric domain names are sometimes labeled as malicious or benign with very low confidence, since the cyber security community have not yet examined these domains rigorously. Particularly, in real-time environments with massive volume of domain names, as considered in this research, new domain names appear frequently. As a result, at each performance evaluation stage, new labeling must be performed for all the domain names. Therefore, most of the labeling in this research is performed automatically and some performed manually by cyber security researchers.

Both methods, automatic and manual labeling, require significant resources. Automatic labeling requires paying for third-party services, while manual labeling requires investing time by cyber security analysts. In addition, manual labeling typically takes a few minutes per domain and therefore, is much slower as compered to automatic labeling that typically takes a few seconds. When labeling with high confidence is required for a massive number of domains, a good approach is to first perform automatic labeling for all domains, and then perform manual labeling for those with low confidence. It is worth noting that in some cases labeling using both methods does not yield a conclusive result. In those cases, the domain is considered as benign, since most of the domains are benign.

In this research, automatic labeling is based on a combination of three third-party services and white/black lists created by a manual inspection during the research period. The first service is Alexa top 20, 000 global sites list (accessed on September 7, 2018) [30] which is used as a white list. The second is Whois registrant information, which provides us with the domain age, the last record update date, the domain current expiration date, etc. The third is Virus Total, a service that uses a collaboration between leading companies in the cyber security industry, researchers, and end users of all kinds, to gather reports about suspicious contents of domain names. The reports contain leading cyber security company analyses of URLs and files, users reports, and domain categories according to other third party services. Once all the information about a domain name is gathered, a statistical analysis is performed and used to label the domain. The automatic labeling produces three labels: malicious, benign and unknown. Domains which were labeled as unknown have not been used for training the algorithm.

In order to report the most up-to-date results, two cyber security researchers have invested 30 full-time days to label up-to-date real data at IBM cyber security lab. This period of time allowed us to label roughly 300,000 domains. This significant amount of domains allowed us to reliably validate the algorithm performance, as well as meet the constraints on human-resources in this collaborative project between Ben-Gurion University and IBM. The manual labeling process was made in a few main stages. The first is gathering additional information about the domain from XForce Exchange [31] and BrightCloud [32]. The second is an analysis of this information combined with the information gathered by the automatic labeling. The third is a web search for malicious activities correlated with the domain and inspection of the domain web site if it exists. These stages produce enough information for the researchers to label the domain.

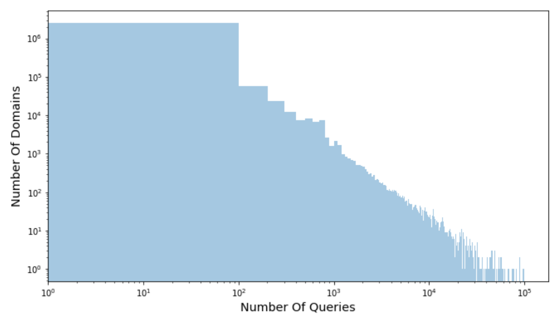


FIGURE 3. The histogram of an average hourly query count per domain in log scale.

B. EXPERIMENTAL RESULTS

We started by examining the quantization division. We first implemented the Equal-Width quantization to negate this intuitive approach. In Equal-Width quantization, the level ranges are uniformly distributed (i.e., $100-200$ queries for level 1, $200-300$ for level 2, and so on). This setting was not performed well for mainly two reasons. First, in terms of DNS queries, the difference between 100 and 200 queries is more meaningful than the difference between $100,000$ to $100,100$ queries, which is not reflected by using the Equal-Width approach. Second, we observed that the nature of the DNS data is unbalanced, i.e. a large number of the domains have very few queries as can be seen in Fig. 3. Thus, using Equal-Width quantization yields an unbalanced histogram, where most of the domains are in the first few quantization levels.

The second intuitive approach that we implemented is the Equal-Frequency quantization. In this approach, each level contains the same number of domains. This method was not performed well due to the distribution of the dataset. Specifically, low level ranges resulted in extremely small values, whereas the high level ranges resulted in extremely large values. Finally, we observed that the Adaptive-Width quantization approach, as described in Section III-A1 performed the best. The Adaptive-Width approach keeps the range in the lower levels relatively small, and in the higher levels relatively large, while keeping reasonable range values which fit the DNS dataset.

We set the target recall rate to $R_t = 80\%$, and set the suspicious domain threshold to detect the spikes accordingly. The threshold was calibrated over the training data until satisfying the constraint. We calibrated the tuning parameters of the DomainObserver algorithm using the same training data as well. Finally, the Random Forest hyper-parameters in REMaDD were chosen. The classifier contains 100 decision trees, where the tree depths are bounded to ten nodes. We optimized it with the Gini impurity criteria and the desired features, as described in Section III-B.

We tested the algorithm on real-time system at IBM cyber security lab using real live streaming data, as described in Section II. For performance evaluations, at the beginning of each test we cleared the B/W lists, such that no prior knowledge from previous tests is used. In Table 2 we present a representative illustration of the SLD data collection. The streaming data was first filtered using the white and black lists. The relevant domains were then passed into the peak detector. In Fig. 4 we present a representative illustration of the peak effect as explained in Section I-A. The suspicious domains were checked for record existence. Non-existent domains were considered as malicious, and existent domains were fed into the deeper inspection using the Random Forest classification method. The deeper inspection performed a feature extraction process, which used additional raw data from the DNS queries database, as can be seen in Table 3. The classification process by the deeper inspection phase is the final classification step, after which a domain was either considered malicious or benign.

TABLE 2. Representative SLD count database.

SLD	Date	Counts
google.com	2019-06-05 12:00:00	676720
facebook.com	2019-06-05 12:00:00	178998
whatsapp.net	2019-06-05 12:00:00	77356
dropbox.com	2019-06-05 12:00:00	59949
*disorderstatus.ru	2019-06-05 12:00:00	3186
microsoft.com	2019-06-05 12:00:00	184424
amazonaws.com	2019-06-05 12:00:00	253590
akamaiedge.net	2019-06-05 12:00:00	558604
*fmniltb.com	2019-06-05 12:00:00	684
huji.ac.il	2019-06-05 12:00:00	264730
akadns.net	2019-06-05 12:00:00	292264
google.com	2019-06-05 13:00:00	584970
facebook.com	2019-06-05 13:00:00	153029
whatsapp.net	2019-06-05 13:00:00	68820
dropbox.com	2019-06-05 13:00:00	55221
microsoft.com	2019-06-05 13:00:00	159423
*disorderstatus.ru	2019-06-05 13:00:00	2851
amazonaws.com	2019-06-05 13:00:00	230508
akamaiedge.net	2019-06-05 13:00:00	499405
huji.ac.il	2019-06-05 13:00:00	231104
akadns.net	2019-06-05 13:00:00	277708
*fmniltb.com	2019-06-05 13:00:00	703

*Malicious domains.

We ran the algorithm over two intervals of two weeks duration each. The first experiment was conducted between December 22, 2018 and January 6, 2019. During this period of time, $74,345$ unique domains ($72,583$ benign, and $1,762$ malicious) passed through our detection system. The second experiment was conducted between February 18, 2019 and March 3, 2019. In this test, $76,461$ unique domains ($74,603$ benign, and $1,858$ malicious) passed through our detection system. The data for training was taken from past measurements offline. We used $70/30$ data splitting for training and testing. We split the tests to validate the results in completely different time frames. During the experiments, the data from IUCC was collected and analyzed every one hour. The results of both intervals were very similar. Thus, we present the average over the two time intervals. The resource consumption was monitored during the run-time to confirm the validation

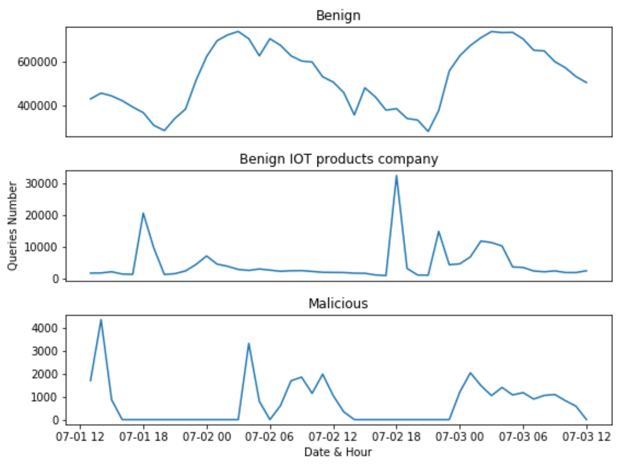


FIGURE 4. A representative illustration of the peak effect of the domain access patterns, in benign, benign IOT companies, and malicious domains.

TABLE 3. Representative DNS raw data.

SLD	FQDN	Q Type	ANS
awsdns-63.net	ns-1019.awsdns-63.net	28	T
facebook.com	z-m.c10r.facebook.com	1	T
*disorderstatus.ru	disorderstatus.ru	1	T
google.com	chat-pa.clients6.google.com	1	T
weizmann.ac.il	ibwsepv02.weizmann.ac.il	1	F
biu.ac.il	commix.cc.biu.ac.il	1	T
*76236osm1.ru	76236osm1.ru	28	T
huji.ac.il	projpc1.fh.huji.ac.il	6	T
mcafee.com	policyorchestration.ccs.mcafee.com	1	T
haifa.ac.il	univ.haifa.ac.il	15	T
*m24.fywkuzp.ru	fywkuzp.ru	1	T
*m25.fywkuzp.ru	fywkuzp.ru	1	T
jpost.com	www.jpost.com	1	T
microsoft.com	teredo.ipv6.microsoft.com	1	F
one.co.il	www.one.co.il	1	T
whatsapp.com	dyn.web.whatsapp.com	28	T
*bbdaowk.com	m2.bbdaowk.com	1	F
mami.gov.il	mami.gov.il	1	F
twitter.com	platform.twitter.com	5	T
icute.co.il	www.icute.co.il	1	F

Notes:
1. *Malicious domains.
2. The presented DNS records were queried at 2019-06-05 12:12:01.
3. Query answers are not presented.

of the constraints as described in Section II-C. The measurements showed that REMaDD used up to 100MB memory at all times. The scanning time was less than one hour as required, where usually it was 35 minutes. The average processing time was 0.5 minute for the W/B list, 6 minutes for the peak detector, 2.5 minutes for the existence check, and 26 minutes for the RFC. The computational complexity was indeed $O(N)$ since the calculations per domain is constant.

Table 4 presents a detailed comparison between the following algorithms: (i) The proposed REMaDD algorithm; (ii) experimental results of REMaDD with Naive Bayes, 5 Nearest Neighbors (5NN) Classifier [33], and AdaBoost, instead of the Random Forest classification in the deeper inspection phase, dubbed REMaDD-NB, REMaDD-5NN, REMaDD-AdaBoost, respectively; (iii) the DomainObserver

TABLE 4. Performance evaluation.

Algorithm	$P(\%)$	$R(\%)$	$A(\%)$	$F_1(\%)$	$P_{TN}(\%)$	$T(s)$
REMaDD	89.4	82.9	99	86.8	99.7	3
REMaDD-NB	86.5	81.1	99	83.78	99.68	3
REMaDD-5NN	85.6	80.6	99	83.1	99.6	3
REMaDD-AdaBoost	86.8	81.3	99	84.1	99.7	3
DomainObserver	76.7	81.7	89.1	79.1	91.3	92
LSTM.MI	67.2	75.3	84.6	71	88	1

algorithm [7] (with a one-hour window size to meet the system constraint); and (iv) the LSTM.MI algorithm [20].

The boldfaced row presents the results obtained by the proposed REMaDD algorithm. The second column presents the precision rate, the third presents the recall rate, and the seventh presents the average running time per domain. We presented three more common statistical measures in the fourth, fifth, and sixth columns, namely the accuracy: $A = (TP + TN)/(TP + FP + TN + FN)$, F_1 score: $F_1 = 2 \cdot P \cdot R/(P + R)$, and true negative rate: $P_{TN} = TN/(TN + FP)$. It should be noted that the dataset used in our real-time system is different from the closed datasets used in [7]. In our experiments, the time windows were significantly larger and the number of inspected domains was significantly larger, as was forced by the real-time system.

Note that REMaDD controls the false positive and false negative rates via the peak detector and RFC. Decreasing the lower threshold for the peak detector decreases the false negative rate, and increases the false positive rate, as typically done in threshold-based detectors. Another way of controlling the trade-off between these errors is by changing the RFC threshold. Setting the RFC in favor of the malicious domain detection improves the true positive rate with the price of decreasing the true negative rate.

As can be seen in Table 4, the proposed REMaDD algorithm that uses the Random Forest classifier in the deeper inspection phase outperformed all other algorithms. The recall met the recall target rate, and the precision was significantly higher than the precision achieved by the DomainObserver and LSTM.MI algorithms. Another interesting result is that the DomainObserver achieved a low detection rate of benign domains as compared to REMaDD performance. The difference between the benign detection rate in [7] and in our setting is due to the percentage of the benign domains in the dataset. While in our dataset the benign/malicious distribution represented real live data, the dataset used in [7] was mostly malicious, which does not represent typical real-world scenarios.

It can be seen in Table 4 that LSTM.MI achieved the worst score in all detection evaluation metrics. These poor results are expected, since LSTM.MI is type-specific and was designed to detect only DGA domain names. Despite its poor performance in these experiments, it achieved the best running time.

Another parameter that we examined was the prediction time per domain, i.e., the time it takes the algorithm to label the domain given all the required data. It can be seen that REMaDD is much faster than DomainObserver (3 seconds under REMaDD as compared to 92 seconds under DomainObserver). This result is achieved due to the REMaDD design of passing the domains through the pre-filtering black/white list and then through the peak detector phase. Furthermore, a Random Forest classifier with 100 decision trees with a maximum of 10 nodes each is lighter then the DTW used by the DomainObserver.

Finally, we present the number of unique domains that passed through each stage of REMaDD, and the average performance achieved by the peak detector, and the existence check and RFC. A total number of 150, 806 unique domains passed through the W/B list and the peak detector. The peak detector achieved the following average scores: $P = 0.38$, $R = 0.89$, $F_1 = 0.53$, $A = 0.96$. These results show that the peak detector performs the crude classification well. The high recall score ensures that most of the domains in the malicious group continue to the next stage. The high accuracy score ensures that most of the domains in the benign group are filtered before the next stage. The low precision and F_1 scores are expected, since the main purpose of the peak detector is to filter out benign domains rather than detect malicious domains. The existence check phase and the RFC have classified 8, 573, and 5, 927 unique domains, respectively. This stage achieved the following average scores: $P = 0.89$, $R = 0.92$, $F_1 = 0.90$, $A = 0.93$. These results show that this stage provides a good classifier for domains with a peak in their access count time series.

V. CONCLUSION

DNS is one of the backbones of the internet, as it maintains a mapping between IP addresses of Internet resources to their corresponding domain names. For this reason, DNS has become an obvious and elementary service in every benign internet related software. On the negative side, it has commonly been abused by criminals for executing malicious activities.

In this paper, we introduced the REMaDD Algorithm, a light weight and computational-efficient real-time algorithm for detecting malicious domains in large-scale networks. We showed a correlation between malicious domains and spikes in their access patterns. Our experimental results showed that our approach works well on the production environment. We compared REMaDD with the DomainObserver algorithm, which was claimed to achieve the state-of-the-art performance in time-series malicious domain detection. We showed that REMaDD outperformed DomainObserver in all tested metrics, particularly in the precision, benign detection rate, and prediction time.

Despite its strong performance, REMaDD has a number of limitations as well. First, it requires a large amount of network traffic data to perform well, which makes it efficient for mainly large-scale networks. Second, REMaDD cannot effectively detect sophisticated domain hijacking attacks, which use benign domains to disguise the malicious activity. These attacks use legitimate domains, and must be treated by detection algorithms for domain hijacking.

As a future work, additional features based on geographic location, TTL, and domain name should be examined to improve REMaDD performance under restrictions on the computational complexity of the feature extraction and data processing in real-time. In addition, in [22] the authors proposed a new labeling method based on heuristics, which provides an easy and fast way for weak (i.e., low confidence) domain labeling. The authors showed that adding this weak labeled data set to the training set improves the classification performance. Adding weakly labeled domains to the training set in REMaDD's RFC module should be investigated for future developments.

REFERENCES

[1] P. Mockapetris and K. J. Dunlap, "Development of the domain name system," in *Proc. Symp. Commun. Archit. Protocols*, vol. 18, 1988, pp. 123–133.

[2] M. Stevanovic, K. Revsbech, J. M. Pedersen, R. Sharp, and C. D. Jensen, "A collaborative approach to botnet protection," in *Proc. Int. Conf. Availability, Rel., Secur.* Berlin, Germany: Springer, 2012, pp. 624–638.

[3] S. Torabi, A. Boukhtouta, C. Assi, and M. Debbabi, "Detecting Internet abuse by analyzing passive DNS traffic: A survey of implemented systems," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3389–3415, 2018.

[4] McAfee. (Feb. 2018). *Economic Impact of Cybercrime*. [Online]. Available: https://www.mcafee.com/enterprise/en-us/solutions/lp/economics-cybercrime.html

[5] H. R. Zeidanloo, M. J. Z. Shooshtari, P. V. Amoli, M. Safari, and M. Zamani, "A taxonomy of botnet detection techniques," in *Proc. 3rd Int. Conf. Comput. Sci. Inf. Technol.*, vol. 2, Jul. 2010, pp. 158–162.

[6] A. Ramachandran, N. Feamster, and D. Dagon, "Revealing botnet membership using DNSBL counter-intelligence," in *Proc. SRUTI*, vol. 6, 2006, pp. 49–54.

[7] G. Tan, P. Zhang, Q. Liu, X. Liu, and C. Zhu, "Domainobserver: A lightweight solution for detecting malicious domains based on dynamic time warping," in *Proc. Int. Conf. Comput. Sci. (ICCS)*. Cham, Switzerland: Springer, 2018, pp. 208–220.

[8] E. Stinson and J. C. Mitchell, "Characterizing bots' remote control behavior," in *Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment*. Berlin, Germany: Springer, 2007, pp. 89–108.

[9] H. Choi, H. Lee, H. Lee, and H. Kim, "Botnet detection by monitoring group activities in DNS traffic," in *Proc. 7th IEEE Int. Conf. Comput. Inf. Technol. (CIT)*, Oct. 2007, pp. 715–720.

[10] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "Exposure: Finding malicious domains using passive DNS analysis," in *Proc. NDSS*, 2011, pp. 1–17.

[11] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "Exposure: A passive DNS analysis service to detect and report malicious domains," *ACM Trans. Inf. Syst. Secur.*, vol. 16, no. 4, p. 14, 2014.

[12] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, "Phoenix: DGA-based botnet tracking and intelligence," in *Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment*, 2014, pp. 192–211.

[13] S. Yadav and A. N. Reddy, "Winning with DNS failures: Strategies for faster botnet detection," in *Proc. Int. Conf. Secur. Privacy Commun. Syst.* Berlin, Germany: Springer, 2011, pp. 446–459.

[14] H. Choi, H. Lee, and H. Kim, "BotGAD: Detecting botnets by capturing group activities in network traffic," in *Proc. 4th Int. ICST Conf. Commun. Syst. Softw. Middleware*, 2009, p. 2.

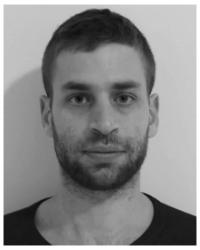
[15] N. Jiang, J. Cao, Y. Jin, L. E. Li, and Z.-L. Zhang, "Identifying suspicious activities through DNS failure graph analysis," in *Proc. 18th IEEE Int. Conf. Netw. Protocols*, Oct. 2010, pp. 144–153.

[16] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning DDoS detection for consumer Internet of Things devices," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2018, pp. 29–35.

[17] B. Rahbarinia, R. Perdisci, and M. Antonakakis, "Segugio: Efficient behavior-based tracking of malware-control domains in large ISP networks," in *Proc. 45th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2015, pp. 403–414.

[18] X.-D. Zang, J. Gong, S.-H. Mo, A. Jakalan, and D.-L. Ding, "Identifying fast-flux botnet with AGD names at the upper DNS hierarchy," *IEEE Access*, vol. 6, pp. 69713–69727, 2018.
[19] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Predicting domain generation algorithms with long short-term memory networks," 2016, *arXiv:1611.00791*. [Online]. Available: http://arxiv.org/abs/1611.00791
[20] D. Tran, H. Mac, V. Tong, H. A. Tran, and L. G. Nguyen, "A LSTM based framework for handling multiclass imbalance in DGA botnet detection," *Neurocomputing*, vol. 275, pp. 2401–2413, Jan. 2018.
[21] S. Schüppen, D. Teubert, P. Herrmann, and U. Meyer, "FANCI: Feature-based automated NXDomain classification and intelligence," in *Proc. 27th USENIX Secur. Symp. (USENIX Secur.)*, 2018, pp. 1165–1181.
[22] B. Yu, J. Pan, D. Gray, J. Hu, C. Choudhary, A. C. A. Nascimento, and M. De Cock, "Weakly supervised deep learning for the detection of domain generation algorithms," *IEEE Access*, vol. 7, pp. 51542–51556, 2019.
[23] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a dynamic reputation system for DNS," in *Proc. USENIX Secur. Symp.*, 2010, pp. 273–290.
[24] I. Ghafir, V. Prenosil, M. Hammoudeh, T. Baker, S. Jabbar, S. Khalid, and S. Jaf, "BotDet: A system for real time botnet command and control traffic detection," *IEEE Access*, vol. 6, pp. 38947–38958, 2018.
[25] B. Sun, M. Akiyama, T. Yagi, M. Hatada, and T. Mori, "AutoBLG: Automatic URL blacklist generator using search space expansion and filters," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2015, pp. 625–631.
[26] Y. Zhauniarovich, I. Khalil, T. Yu, and M. Dacier, "A survey on malicious domains detection through DNS data analysis," *ACM Comput. Surveys*, vol. 51, no. 4, pp. 1–36, Jul. 2018.
[27] S. Shin, Z. Xu, and G. Gu, "EFFORT: Efficient and effective bot malware detection," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 2846–2850.
[28] K. Alieyan, A. Almomani, A. Manasrah, and M. M. Kadhum, "A survey of botnet detection based on DNS," *Neural Comput. Appl.*, vol. 28, no. 7, pp. 1541–1558, Jul. 2017.
[29] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[30] Amazon Web Services. *Alexa Top 1M Sites*. Accessed: Sep. 7,'2018. [Online]. Available: https://aws.amazon.com/alexa-top-sites/
[31] *IBM X-Force Exchange*. Accessed: 2019. [Online]. Available: https://exchange.xforce.ibmcloud.com
[32] *Webroot BrightCloud*. Accessed: 2019. [Online]. Available: https://www.brightcloud.com/
[33] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.

OFIR ERETS KDOSHA received the B.Sc. degree in electrical and computer engineering from Ben-Gurion University, Israel, in 2011, where he is currently pursuing the M.Sc. degree in electrical and computer engineering, under the supervision of Dr. K. Cohen. He has been working as a Data Scientist with the IBM's Cybersecurity Centre of Excellence (CCoE), Beer-Sheva, Israel, for the last two years. His current research interests include cyber-security, machine leaning, and data science.



GILAD ROSENTHAL received the B.Sc. degree in electrical engineering from Ben-Gurion University, Israel, in 2017, where he is currently pursuing the M.Sc. degree in electrical and computer engineering, under the supervision of Dr. K. Cohen. He has worked as a Data Scientist at both start-ups and corporates. His current research interests include machine learning, artificial intelligence, deep learning, anomaly detection, and data science in cybersecurity domain.

KOBI COHEN received the B.Sc. and Ph.D. degrees in electrical engineering from Bar-Ilan University, Ramat Gan, Israel, in 2007 and 2013, respectively. In October 2015, he joined the Department of Electrical and Computer Engineering, Ben-Gurion University of the Negev (BGU), Beer Sheva, Israel, as a Senior Lecturer (also known as Assistant Professor in USA). He is also a member of the Cyber Security Research Center and the Data Science Research Center, BGU. Before joining BGU, he was as a Postdoctoral Research Associate with the Coordinated Science Laboratory, University of Illinois at Urbana–Champaign, from August 2014 to July 2015, and the Department of Electrical and Computer Engineering, University of California at Davis, Davis, from November 2012 to July 2014. His main research interests include decision theory, stochastic optimization, and statistical inference and learning, with applications in large-scale systems, cyber systems, wireless, and wireline networks. He received several awards, including the Best Paper Award in the International Symposium on Modeling and Optimization in Mobile, Ad hoc and Wireless Networks (WiOpt) 2015, the Feder Family Award (second prize), granted by the Advanced Communication Center at Tel Aviv University, in 2011, the President Fellowship, from 2008 to 2012, and top honor list's prizes from Bar-Ilan University, in 2006, 2010, and 2011.

ALON FREUND received the B.Sc. degree from the Communication Systems Engineering Department, Ben-Gurion University. He is currently pursuing the M.Sc. degree with the Software and Information Systems Engineering Department, Ben-Gurion University. He is currently working with the Cyber Security Center of Excellence (CCoE), Beer Sheva, Israel. His main fields of interests include network security and data science.

AVISHAY BARTIK received the B.Sc. degree in mathematics and computer science from the Open University of Israel, in 2010. He is currently a Security Researcher with the IBM's Cyber Security Center of Excellence, Beer-Sheva, working on various aspects of networks and system security. Prior to joining IBM, he has served as a Security Software Engineer with PMO.

AVIV RON received the B.Sc. degree in computer science from Ben-Gurion University, Israel, in 2007. He has been working as a Software Engineer with Intel for five years, a Security Researcher and an Architect with Intel for four years, and a Senior Security Researcher with IBM for five years. He also served as an External Lecturer on the topic of cyber security with Ben Gurion University, for four years. He has 17 patents. His current research interest includes detecting cyber threats by applying artificial intelligence.