ARBA: Anomaly and Reputation Based Approach for Detecting Infected IoT Devices

GILAD ROSENTHAL, OFIR ERETS KDOSHA, KOBI COHEN, ALON FREUND, AVISHAY BARTIK, AND AVIV RON

1School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Be’er Sheva 84105-01, Israel
2IBM Cyber Security Center of Excellence, Be’er Sheva 84439-44, Israel

ABSTRACT

Today, cyber attacks are constantly evolving and changing, which makes them harder to detect. In particular, detecting attacks in large-scale networks is very challenging because they require high detection rates under real-time resource constraints. In this paper, we focus on detecting infected Internet of Things (IoT) hosts from domain name system (DNS) traffic data. IoT hosts, such as streaming cameras, printers, air conditioners, are hard to protect, unlike PCs and servers. Enterprises are often unaware of the devices which are connected to the network, their types, makes, and vulnerabilities. Since IoT hosts make use of the DNS protocol, analyzing DNS data can give a broad view of malicious activities, because they abuse the DNS protocol and leave fingerprints as part of their attack vector. In this collaborative research between Ben-Gurion University, and IBM, we establish a novel algorithm to detect infected IoT hosts in large-scale DNS traffic, named Anomaly and Reputation Based Algorithm (ARBA). Its novelty resides in developing a framework that combines host classification and domain reputation in a real-time production environment. ARBA is highly computational efficient and meets real-time requirements in terms of run time and computational complexity. By contrast to existing algorithms, it does not require a massive traffic volume for training, which is of significant interest in detecting infected hosts in real-time. The research was conducted on real live streaming data from IBM internal network traffic, and confirm the algorithm’s strong performance in a real-time production environment.

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

I. INTRODUCTION

It is amply clear today that the internet is often exploited by cyber-attackers against different targets. As a result, malicious contents are spread over the internet, but are mostly hidden or disguised as benign services. Since today’s society relies heavily on the internet, ways of keeping it safe for users are more crucial than ever. Cyber researchers are in a never-ending battle to detect malicious activities to prevent, mitigate, and block them.

This article focuses on detecting infected IoT hosts. Many devices that were considered dumb not long ago are now connected to the network and able to report their state, as well as receive instructions and firmware updates from the cloud (see [1]–[3] for recent developments and applications). However, IoT devices present a risk to enterprise networks. For financial reasons, most smart connected devices usually have limited computing resources and hence run limited versions of operating systems (OS). In addition, the range of devices and OS is considerable. Thus, IoT devices are hard to manage, unlike PCs and servers which run a common small set of OS that can be managed by agents. The typical risk scenario occurs when an employee brings an un-managed IoT device to work and simply plugs into the network, thus posing a threat to the enterprise.

Major concerns in IoT applications include data privacy, and security challenges, since IoT technology allows data to be transferred seamlessly from surveillance devices to the internet. The heterogeneity and complexity of IoT networks, the fact that IoT devices lack common standards, and the wide
variety of devices, types, and operating systems, add significant challenges for securing them. Furthermore, the devices are usually designed to run minimal software operations on a cheap hardware. This comes at the expense of security risks. For example, running protection software such as antivirus on an IoT device is often impractical.

A common method of launching attacks on IoT devices is via a botnet. The attacker gains control of the compromised IoT devices and use them for its own means (e.g., to launch distributed denial of service (DDoS) attacks, or steal data). A well-known example of this attack is the Mirai malware that emerged in 2016. The attacker initially took control of consumer devices such as routers and webcams to launch DDoS attacks. When an IoT device is compromised, its DNS behavior changes abruptly. It operates communications with new domains, its command and control (C&C) servers, and sends new and different queries from those usually sent. The suggested ARBA algorithm exploits these traffic fingerprints and aims to detect the infection by examining the host DNS traffic behavior. For example, Mirai DDoS attack on Dyn DNS provider could be detected by ARBA which analyzes the abrupt change in the DNS behavior of the printers used in the attack. Other types of attacks that affect the DNS traffic, such as Reaper (a.k.a. IoTroop) and Echobot, can be detected by ARBA as well. Since IoT hosts make use of the Domain Name System (DNS) protocol, this article looks specifically at ways to detect infected IoT hosts from DNS data.

DNS is one of the fundamental protocols of the internet. It is responsible for mapping IP addresses to domain names and vice versa, and acts as the internet “phone book”. The distributed nature of the system, and the fact that DNS is widely used, and typically open and unfiltered, are key advantages for malware developers. Nevertheless, since malicious DNS queries leave fingerprints that can be monitored, one of the most promising directions in internet cyber security research efforts remains the analysis of DNS [4].

Malicious hosts are typically a part of a botnet controlled by C&C which affects the DNS pattern of the hosts. Therefore, malicious hosts change their DNS pattern abruptly at the moment of infection, differing from benign IoT hosts’ DNS pattern that remains roughly the same over time.

Detecting malicious activities over DNS in form of malicious domains or infected hosts is a very challenging task. On top of ransomware, botnets, phishing, and domain generation algorithm (DGA), cyber criminals continuously design new types of attacks and evasion techniques. Since cyber crimes are very profitable, the crime industry has become increasingly sophisticated. Malicious activity techniques are constantly evolving and changing, thus making them harder to detect [5]. Detection is even harder when it needs to be executed on large-scale networks, given the amount of data that must be analyzed. Since analyzing every host or domain name in depth is impractical for computational reasons, computationally efficient detection algorithms to analyze DNS traffic with high reliability need to be developed. In addition, using passive traffic measurements (i.e., measurements that were derived from the network traffic passively, without creating or modifying the network traffic) is advantageous in detection algorithms, since the network remains in its true pattern state, and overloading the network is avoided. Here we address these issues by developing a lightweight algorithm for detecting IoT infected hosts in large-scale networks based on passive DNS measurements.

A. MAIN RESULTS

Studies of infected machines and intrusion detection via DNS have mostly tackled the problem by using knowledge-based algorithms or machine learning-based algorithms [4], [6]. Knowledge-based algorithms are only able to detect known infections through signature matching. This is done mostly by searching for known patterns that characterize a specific botnet behavior or by using continuously updated blacklists such as DNS-based blacklist (DNSBL) [7]. These methods have successfully been applied for detecting specific attacks with known characteristics. However, they tend to fail when required to detect new malicious activities in large-scale networks. Offline machine learning-based algorithms usually suffer from substantial disadvantages that create difficulties when attempting to apply them to real-time large-scale networks. First, the training phase of such algorithms requires labeled data which are hard to acquire. Second, these algorithms tend to have many false positives which can not be tolerated in real network security applications. In addition, the memory usage in these algorithms is too demanding since a large amount of data is required for processing. They also suffer from scalability issues, especially in large-scale networks [8], which is the focus of this study. In addition, most of these algorithms were developed in a “clean” environment, using closed, and sometimes even synthetic datasets. Therefore, their performance on real data in real-time systems can be considered unclear. Below, we summarize our main contributions:

1) A Novel Framework for Detecting Anomalous hosts

Using DNS: We developed a novel framework for detecting infected hosts using DNS data. The novelty resides in incorporating host analysis with domain reputation analysis. This constitutes a new detection framework that combines knowledge-based and ML-based components for both hosts and domains. Furthermore, we develop an anomaly detection method instead of traditional host clustering which decreases the amount of data needed to detect infected machines. It uses ML-based analysis of DNS features in a real-time environment.

2) Real Data Analysis: We analyzed infected and benign host DNS query patterns based on real live streaming passive traffic measurements. Abrupt changes in the hosts’ DNS patterns were observed in various forms of cyber-attacks. Our experiments validate the supposition that there is a correlation between deviations in the behavior of a host’s DNS traffic and its infected state. Using real data analysis enabled us to discover and extract meaningful DNS features for detecting anomalous hosts. These features were used for the algorithm.
development. For example, we observed that the communication probability between a host’s country and a resolved IP in a certain country provides significant information for detecting malicious hosts, as illustrated in Figure 3 in Section IV.

3) Algorithm development: Most ML-based algorithms for classifying DNS infections are supervised, and require a massive amount of labeled data to train the algorithm, which are hard to acquire (see discussion in IV.A.). By contrast, we propose a new approach for detecting host infections over DNS that does not require labeled data, by using an anomaly detection module. Our approach is based on anomaly detection and domain reputation engines when detecting infected hosts. By looking at each host data as a separate time series, the proposed Anomaly and Reputation-Based Algorithm (ARBA) detects infected hosts even with just a few samples. The only supervised module in our system is the domain reputation engine, which is trained separately, based on different samples from the same training distribution. ARBA is scalable and computationally efficient, and is used to detect infected hosts in real-time large-scale networks.

ARBA works as follows. First, it implements low-complexity filtering of the live streaming data by performing domain lists analysis through similarity measures. This stage declares suspects quickly and significantly reduces the number of tested hosts for the next deeper inspection phase. Then, it performs feature extraction of the remaining filtered hosts followed by anomaly detection algorithm, Isolation Forest. These steps allow the algorithm to run the deeper inspection phase with very low computational consumption, run time, and memory usage. Then, we develop a real-time deep inspection domain classifier to label the domains as malicious or benign using a custom real-time Random Forest classification.

4) Extensive Performance Evaluation in a Real-Time System: Enterprises tend to be reluctant to share their data and real-time cyber security performance because of the legal risks of violating privacy, or to avoid sharing information that could benefit their competitors. Therefore, analyzing and validating cyber security algorithms in real-time systems using up-to-date real data is one of the main hurdles in academic cyber security research. This collaborative research between Ben-Gurion University, and IBM constitutes an important contribution in this respect. Specifically, it enabled us to analyze and validate the algorithm’s performance in a real-time environment using up-to-date live streaming data. To evaluate the algorithm’s performance, we deployed it in a real working environment (see Section II for details). Note that labeling hosts and malicious domains 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. Here, two cyber security researchers invested a significant amount of time (over several months during this project) to label the up-to-date real data of IoT hosts and domain names in the data (see Section IV.A.).

The real-time detection performance and resource consumption achieved by ARBA were satisfying and highly compatible in all tests, and outperformed BotDAD. Furthermore, we deployed ARBA in a real-time production environment on the IBM network, and demonstrated its ability to detect and report real-world infected hosts in two different scenarios in real-time. To the best of our knowledge, ARBA is the first algorithm in the academic cyber security literature that presents strong performance in the detection of infected hosts in a real-time production environment.

B. RELATED WORK

DNS analysis presents an efficient method in IoT security, and a good overview can be found in [9], [10]. Device and software system implementations that monitor the traffic generated by IoT devices was suggested in [11]. An attack detection approach based on machine learning for anomaly detection in IoT environments was studied in [12].

1) KNOWLEDGE-BASED AND MACHINE LEARNING-BASED APPROACHES

As overviewed in [4], most of the malicious activity detection methods via DNS are either knowledge-based, or ML-based. The latter approach is more common these days due to its robustness. Knowledge-based methods [13]–[18] rely on expert insights obtained through measurements and studies. For instance, in [13], the authors proposed a botnet detection mechanism by monitoring passive DNS traffic, which correlates group activity in DNS queries simultaneously. This research was based on the notion that a botnet usually has a fixed size of hosts that access a domain name, and appear and disappear intermittently. This approach can lead to a high false positive rate for some IoT hosts due to version updates which usually behave similarly to botnet behavior. A more recent work used blacklists to perform cause-based classification of malicious DNS queries [19].

Knowledge-based methods have several limitations. First, cyber security researchers sometimes tend to be biased, unintentionally or intentionally [4]. Second, this type of solutions is problem-specific, which limits their robustness, as compared to ML-based methods. For example, in [20], the authors developed DBod, a system for detection and classification of infected machines based on statistical similarity between query behaviors. Although DBod was reportedly accurate and effective, it is limited to malware that uses DGA and cannot be effectively adapted to handle other types of malware. Another limitation is the difficulty of analyzing high-dimensional data. These limitations triggered the need to develop ML-based methods for detecting malicious activities.

ML-based techniques for malicious activity over DNS can be broadly classified into two main groups. The first observes the host side, the hosts’ behavior, and interaction between hosts in a network [21]–[23]. In [24], the authors developed the PsyBoG algorithm, a scalable method for detecting botnet...
group activity in large-scale networks by scanning periodic patterns of traffic. The scalability issue stems from the use of naive techniques such as signal processing, periodical analyzers, and non-learning techniques. Although PsyBoG meets the memory, processing time, and scalability requirements, it does not perform well in terms of detecting small botnet groups. Specifically, the analysis depends on the behavior of the group of hosts. As a result, instead of examining each host separately, a comparison across hosts and similarity search is needed. This weakens PsyBoG detection performance, since it can only detect a group of infected hosts and cannot detect a single infected host. In addition, it does not use blacklists, which decelerates the algorithm.

In [25], the authors proposed the DomainObserver algorithm for malicious domain classification. DomainObserver applies passive traffic measurements and time series data mining techniques to detect malicious domains. The authors created three types of time series data for each domain: access, users, and entropy. These series were constructed from aggregated traffic measurements for a certain time window. Lastly, a K-Nearest Neighbor (KNN) classifier was applied using Dynamic Time Warping (DTW) as the distance metric. In [26], the authors developed BotDAD, an architecture for detecting bot-infected machines using DNS fingerprints. Their architecture was comprised of two main subsystems: a DNS fingerprinting module, and an anomaly detection module. Similar to our research, in BotDAD, aggregated features of a host are extracted. These features are processed through an anomaly detection engine, and a score is assigned by a classifier. BotDAD resembles our algorithm by using features extracted from DNS for anomaly detection. BotDAD suffers from several disadvantages that our work aims to solve. First, it uses a ML-based method without tapping knowledge-based lists, which decelerates the algorithm. Second, retraining phases are required due to the frequent changes in host behaviors, which limits its applicability in real-time systems. This issue was pointed out as a future research direction in [26]. By contrast, here our method generates and continuously maintains black and white lists, updated using the ARBA algorithm output and outside vendor services such as [27]–[29]. In addition, unlike [26] which combined correlated features from the host and domain sides, our approach analyzes the host side and the domain side separately, which significantly improves detection performance. Specifically, the domain reputation is trained as a stand-alone API and is not trained using the same data as in the anomaly detection engine.

In [23], the authors designed a pipeline to operate on network middle-boxes (e.g. routers and firewalls) to identify anomalous traffic and hosts related to these anomalies. This research used other sources of data in addition to DNS, such as packet sizes, bandwidth information, etc. This type of data, however, are not always available in real applications. In addition, they focused on DDoS attacks specifically. Another problem-specific solution was developed in [30], where the authors proposed a ML framework for identifying and detecting DGA domains based on time-series analysis using a deep neural network.

In [31], the authors developed a method for detecting tunneling and low throughput data exfiltration over DNS. The focus was on detecting and denying requests to domains which are registered for cyber-campaign. First, DNS logs are collected per domain in a manner that permits scanning for long periods of time and intends to handle “low and slow” attacks. Then, a pre-trained one-class classifier is used to detect domains that exchange data over the DNS. Although the method is applicable in large-scale networks, DNS logs do not always include full details of the query or response. As opposed to our method that uses the host DNS pattern, in [31] the analysis is based solely on the domain, while the host DNS pattern is overlooked. In [32], the authors developed a real-time mechanism for detecting exfiltration and tunneling of data in large-scale networks. The authors proposed an anomaly detection algorithm for detecting malicious DNS queries. The system extracts attributes from the domain name which was queried and inputs them to isolation forest to test if a new query is anomalous or not. Similar to the system presented in [31], this method overlooks the host DNS behavior as well. Finally, the method in [33] is used for a fundamentally different purpose, as it aims at detecting only DGA NXdomains. By contrast, we aim at detecting infected hosts (not just NXdomains) using their aggregated features based on queries they have made and not only by classifying the domains these hosts query.

The second group of ML-based techniques for malicious activity detection using DNS deals with the resolution part, the domains, and their resolved IPs. In [34], the authors proposed an approach to detect malicious domains by analyzing a massive mobile web traffic data. In [35], the authors proposed a semi supervised ML model using a neural network to identify anomalies in network traffic. The goal was to detect potential attacks hidden by fast flux. This was done by training a neural network to evaluate the relationship between the domain name and the resolved IP of DNS queries and classify the resolution as benign or malicious. Another example of resolution-based malicious activity detection can be found in [36], where the authors used a Bayesian classifier for features extracted from DNS fingerprints to detect fast flux. Since fast flux is a technique that focuses on domain name and its resolutions, the extracted features are resolution-related and no host information is needed.

Combining knowledge-based and ML-based methods has a great potential to provide a robust solution that can detect different attack vectors, which is the focus of this research.

II. SYSTEM MODEL AND PROBLEM STATEMENT

A. DESCRIPTION OF THE ENVIRONMENT AND PREMISES

Consider the following Internet Service Provider (ISP) DNS architecture as illustrated in Fig. 1. The system consists of C hosts, some of which are connected to R DNS Recursive Resolvers (RRs). When a host queries a domain name,
it addresses the DNS RR. Then, the RR searches the domain name in its cache. If it does not appear, the RR queries the internet root servers to get the name servers for the top level domain (TLD) (e.g., “.com”). Then, it queries the required TLD name servers for the authoritative name server and so on recursively until it reaches an authoritative name server. Finally, it queries the authoritative name servers to get the IP address for the host domain and returns the IP to the host.

In the network architecture, some DNS hosts receive DNS services from the internal enterprise RR, while others receive their DNS service from an external RR. An external RR is a recursive resolver which is located outside the enterprise network, and provides DNS resolutions. For example, Quad9 operates free DNS servers at the IP address 9.9.9.9. The network sniffer is located above the RR level and the hosts’ level. If the RR is located inside the network, the host IP will not be available at the sniffer level. Thus, the sniffer sees all the RR traffic as one host. Therefore, a host that uses internal RR will not be analyzed separately and should be removed in the preprocessing stage. These environment settings define two groups of hosts. In the first, the IPs are known. In the second, the IPs are hidden behind the RR, and therefore the IP addresses of the endpoint hosts are unknown.

Specifically, 98% of the hosts in the network receive their data from external RR, and therefore the sniffer sees the end point traffic. The rest 2% of the hosts in the network receive DNS data from internal RR, where the host IP is hidden behind the RR.

The research and development in this paper rely on the following model premises. First, IoT hosts are mostly characterized by static behavior in terms of number of queries, number of unique second level domains (SLDs) they query, query types, etc. They usually query a limited number of domains, and often the same domains every day [37]. Some hosts query their control server periodically for updates. Second, in the test network, most hosts have a static IP; i.e., their IP does not change for a duration of more than a month. This working assumption was tested and validated in this research. Specifically, we analyzed the network data by using reverse DNS queries, which reveals the hostname and IP mapping, to support our observation. Nevertheless, our development applies to networks in which the mapping between hosts and IPs changes more frequently as well, as long as it is possible to correlate the host to the DNS queries it made. Third, the number of infected hosts in the training phase is assumed to be sufficiently small to allow efficient training (less than 0.5%). Since the initial training is done offline once per months and deeper inspection can be made (in our network, we implemented the initial training once per two months), it is common to assume a clean network in this phase.

We point out that in the IBM network from which the data for this research was collected, the RR was not defined to support DNS over HTTPS (DoH). In addition, when using public RRs that offer DNS solutions, such as Quad9 or Cloudflare, the logs are inaccessible as well. Therefore, we could not retrieve the DNS logs. Nevertheless, when the logs are available, they can be used in the algorithm as well to detect DoH.

B. LIVE STREAMING DATA

The real live streaming data analyzed in this study were the enterprise network of IBM USA. The data were in the form of a real-time passive raw DNS collection. As shown in Fig. 1, a sniffer is placed above the RR level to capture the records. The sniffer collects every DNS query and its response, and stores them in a database. Every record contains the full DNS query and its response data. On average, 74 million new records were collected every day from 2,500 hosts that query more than 1 million unique domains. Among them, 110 were IoT hosts. Due to the massive volume of data, we used a distributed file system, the Hadoop Distributed File System (HDFS).

Due to the requirement of real-time detection under limited computational resources, the massive data size, and the exploitation of the periodic nature of the daily IoT traffic, we used daily aggregated training data to update the classifier. The algorithm classifies hosts in real-time up to the system requirement of a one-hour latency. The entire development of the algorithm was based on up-to-date records from this database. The use of real live streaming data at each stage of the development guarantees that the algorithm meets real-time resource restrictions and performs well (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., an infected host is classified as malicious), False Negative (i.e., an infected host is classified as clean), False Positive (i.e., a clean host is classified as infected) and True Negative (i.e., a clean host is classified as clean)
as clean) binary classification results, respectively. Let
\[ P \triangleq \frac{TP}{TP + FP} \]  
(1)
denote the Precision score, and let
\[ R \triangleq \frac{TP}{TP + FN} \]  
(2)
denote the Recall score. A summary of the notations used in the equations throughout this paper is given in Table 1.

### Table 1. Table of notations.

| Symbol | Description |
|--------|-------------|
| \( P \) | Precision score |
| \( R \) | Recall score |
| \( TP \) | Number of true positives |
| \( FP \) | Number of false positives |
| \( FN \) | Number of false negatives |
| \( N \) | Number of true negatives |
| \( D_{CL} \) | Domains queried by client \( j \) at day \( i \) |
| \( s(x, y) \) | Isolation forest score for observation \( x \) for trees with \( n \) external nodes |
| \( n(x) \) | Path length of observation \( x \) |
| \( P_{CI}(p) \) | For \( CI \) impurity with class distribution vector \( p \) |
| \( P_{CI}(p) \) | Communication probability between host country \( i \) and resolved country \( j \) |
| \( N_{QR} \) | Number of queries between host country \( i \) and resolved country \( j \) |
| \( K \) | Number of trees in the Random Forest |
| \( OPT(x_j) \) | The binary classification score of decision tree \( j \) for observation \( x \) |

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

Thus, the objective was to develop an algorithm that maximizes \( P \) under the constraint that \( R \geq R_t \). In the experiments, we set the target recall rate to \( R_t = 0.85 \), which was dictated by IBM security service requirements and is a typical value in cyber security services.

In addition, the algorithm was destined to run on the IBM production environment in real-time. Therefore, it had to 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 queried by all hosts in the network.

**Constraint C2:** The time complexity of the algorithm for processing the collected data samples must be less than one hour.

### III. THE ANOMALY AND REPUTATION-BASED ALGORITHM (ARBA)

We now present the ARBA algorithm to solve the objective under the constraints in Section II.C. ARBA is designed as a lightweight algorithm to achieve the time complexity constraint. Specifically, we judiciously design ARBA to limit the amount of inference data passing through expensive components to meet the time complexity constraint. Second, we design the anomaly detection engine to be robust to changes in the number of hosts in the network. Finally, achieving a low false positive rate is achieved by judiciously designing threshold-based detection in several components of the system. We divide the system into three main stages, as illustrated in Figure 2. First, the data are preprocessed. In this stage, servers, computers, and cellphones are filtered, leaving only IoT endpoint hosts. Second, an anomaly detection phase for these hosts is performed. In this stage, the similarity between host’s SLD list is checked. Then, if it is unclear whether the host is infected or not by examining the similarity of the list, the host is further analyzed. Third, when a host is suspected of being infected, a score is given to each new domain the host queried by the Domain Reputation Engine (DRE). The DRE is comprised of white and black lists. Then, a random forest classifier produces a score for the domain reputation. Finally, a decision is made whether the host is infected or not. We next describe each component of ARBA in detail.

### A. PREPROCESSING

The purpose of the preprocessing stage is to filter out DNS hosts which are not governed by static behavior. We start by taking the following steps to clean the network from hosts that normally behave in an unpredictable fashion such as personal computers, RR servers, mobile phones, etc., to guarantee that the remaining hosts are endpoints users (not internal RR servers) with static behavior pattern which can be predicted. At first, crude filtering is done based on the daily number of queries and the number of unique SLDs that the host queried. Then, we use indicative domains that determine the identity of a host to filter out the unpredicted hosts. Finally, we validate using heuristics on the host names that no unpredicted hosts remain. Note that the crude filtering step, in which all those hosts are filtered out and are not tested by the next stages of the algorithm, is done offline, once per weeks or months (in our experiments we ran it once per two months). It is not part of the real-time implementation. This step is needed for efficient implementation in real-time since the algorithm is designed to detect infected devices with static DNS patterns, which characterizes a wide range of IoT devices.

### B. ANOMALY DETECTION ENGINE

In anomaly detection, the goal is to find objects that differ from most other objects in a group (see related work in [38]–[48] and references therein). Anomalous objects, also known as outliers, usually lie far away from other data points in the feature space. We develop this module to distinguish between infected and benign hosts. This is done by first comparing the host SLD list queried on two consecutive days. If a certain host’s similarity measure on these two days exceeds a certain threshold, it is suspected of infection. Then, its DNS features are extracted and inputted to an Isolation Forest (IF) algorithm. The IF declares it to be benign or suspected of infection, and passes it onto the next stage of the system.
1) DOMAIN LIST ANALYSIS VIA SIMILARITY MEASURES (DLASM)

The first component of the anomaly detection engine, dubbed DLASM, computes a similarity measure in the host domain lists for two consecutive days. Since our product is designed to detect anomalies in IoT hosts, which typically have static query behavior, each host is expected to query mostly the same domains every day. Based on this insight, we created two lists containing all queried SLDs on two consecutive days for each host in the database, and computed a similarity score between the lists. Mathematically, for each host $j$, denoted by $C_j$, the similarity score, denoted by $S_{C_j}$, is given by:

$$S_{C_j} = \frac{|D_{C_i}^j \cap D_{C_{i+1}}^{i+1}|}{|D_{C_i}^j \cup D_{C_{i+1}}^{i+1}|},$$

where $i, i+1$ denote the indices of the consecutive days $i, i+1$, respectively. The terms $D_{C_i}^j \cap D_{C_{i+1}}^{i+1}$ and $D_{C_i}^j \cup D_{C_{i+1}}^{i+1}$ denote the intersection and union of the list sets $D_{C_i}^j, D_{C_{i+1}}^{i+1}$, with cardinality $|D_{C_i}^j \cap D_{C_{i+1}}^{i+1}|, |D_{C_i}^j \cup D_{C_{i+1}}^{i+1}|$, respectively. We set zero if the sets are empty.

Let $L, U$ be the lower and upper thresholds of the list similarity scores, where $L < U$. The DLASM module works as follows. For each host (say $j$) we compute the similarity score $S_{C_j}$. If it is equal or less than $L$, the host is considered anomalous. If it is equal or greater than $U$, the host is considered benign. If $L < S_{C_j} < U$, further examination is needed to determine whether it is infected or not.

There is a tradeoff between implementation complexity and detection accuracy. Intuitively, the margin between $L$ and $U$ should be small enough to detect hosts in an early stage of the algorithm and reduce the number of hosts passing on to the IF and the DRE. On the other hand, it should be large enough to test hosts with low confidence regarding their state to reduce the error. In the experiments, the thresholds $L, U$, were set to 0.25, 0.8, respectively, which yielded strong performance based on empirical evaluations.

2) HOST FEATURE EXTRACTION

In order to identify the anomalous behavior of a host, an extraction of meaningful DNS features is required. We designed a selective feature extraction to reduce the number of false positives. Some of the features were normalized per host by dividing each of them by the number of total queries made by the host that day. This was done after making an extensive feature analysis by comparing results on the validation set with and without feature normalization. The features are calculated on a daily basis using a sliding window of one hour. Specifically, at each hour we declare the host state, where we use the daily data aggregation to exploit the periodic nature of the daily IoT traffic. We make these aggregations for a week to exploit the periodic nature of the weekly IoT traffic, as well. The selected features are
presented in Table 2. A detailed description of the features is given in the appendix.

**TABLE 2. A feature list for the isolation forest.**

| Features Set               | # Feature Name                                      |
|----------------------------|-----------------------------------------------------|
| Time-related Features      | 1 Number of unique hours of queries                 |
|                            | 2 Number of daytime queries (N)                     |
|                            | 3 Number of nighttime queries (N)                   |
|                            | 4 Number of max queries in 1-minute window          |
|                            | 5 Number of max queries in 1-hour window            |
| Queries-related Features   | 6 Number of unique query types                      |
|                            | 7 Query type (N)                                    |
| Resolution-related Features| 8 Number of unique response codes                   |
|                            | 9 Response code type (N)                            |
| Domain-related Features    | 10 Number of unique domain queried (N)              |
|                            | 11 Number of Alexa domains queried (N)              |
|                            | 12 Minimum queries per domain (N)                   |
|                            | 13 Maximum queries per domain (N)                   |
|                            | 14 Mean of queries per domain (N)                   |
|                            | 15 Standard deviation of queries per domain (N)      |

(N) stands for normalized features.

3) **ISOLATION FOREST**

The features of the suspected anomalous hosts are inputted to an isolation forest (IF) based algorithm. The IF algorithm explicitly identifies anomalies instead of profiling normal data points [49]. Similar to tree ensemble based methods, it is based on decision trees. In these trees, partitions are generated by first randomly selecting a feature, and then selecting a random split value between the minimum and maximum value of the selected feature. Since recursive partitioning can be represented by a tree structure, the number of splits required to isolate a sample is equivalent to the path length from the root node to the terminating node. This path length, averaged over a forest of these random trees, is a measure of normality and was used for our decision function. Note that outliers were less frequent than regular observations in our framework. In addition, they are usually distant from the typical observation embedded in the feature space. Thus, random partitioning produces noticeably shorter paths for anomalies. Hence, when a forest of random trees collectively produces shorter path lengths for specific samples, they are likely to be anomalies.

The anomaly score used in the IF module to indicate whether a host is benign or anomalous was given by:

\[ s(x, n) = 2 - \frac{E(h(x))}{c(n)} \]  

where \( h(x) \) is the path length of observation \( x \), \( E(h(x)) \) is the average of \( h(x) \) from a collection of isolation trees, \( c(n) \) is the average path length of an unsuccessful search in a binary search tree, and \( n \) is the number of external nodes. Each observation is given an anomaly score and the following decision is made. A score close to 1 indicates that the observations are anomalies, and a score less than 0.5 indicates that the observations are normal.

Note that IF is highly computationally efficient, since it does not require complex computations of distance or density measures. Specifically, it enjoys linear time complexity and low memory requirements [49]. Finally, it can scale up to handle extremely large data-size and high-dimensional problems.

**C. THE DOMAIN REPUTATION ENGINE (DRE)**

The DRE is a component that outputs a score (between 0 and 1) to a domain based on the historical pattern of the domain DNS records. It builds models of known legitimate domains and malicious domains, and uses these models to compute a reputation score for a new domain, indicating whether the domain is malicious or legitimate [50]. Specifically, the domain reputation is a score which is calculated for each domain of a suspected host. The reputation is between 0 and 1 and describes a measure of the maliciousness of a domain. Ideally, a score close to 1 indicates a malicious domain, whereas a score close to 0 indicates a benign domain. The reputation is computed by a random forest binary classifier of \( T \) trees \((T = 100 \) in our experiments). The score \( \hat{y}_i \) for observation \( x_i \) is calculated as follows:

\[ \hat{y}_i = \mathbb{1} \left\{ \left( \frac{1}{T} \sum_{j=1}^{T} DT_j(x_i) \right) - 0.5 \geq 0 \right\} \]  

where \( DT_j(x_i) \) denotes the binary classification score of decision tree \( j \) for observation \( x_i \), and \( \mathbb{1} \left\{ \left( \frac{1}{T} \sum_{j=1}^{T} DT_j(x_i) \right) - 0.5 \geq 0 \right\} \) denotes the indicator function, which equals 1 if \( \left( \frac{1}{T} \sum_{j=1}^{T} DT_j(x_i) \right) - 0.5 \geq 0 \) holds, or 0 otherwise.

The aim of the DRE module in ARBA is to assign a reputation score to newly observed domains queried by the suspected anomalous hosts. For each of these hosts we observed the changes in the domain list queried by a host. We created a list of newly observed domains for the host, and gave each domain a score. If the domain reputation is above the threshold, it is declared as malicious. If a host queries malicious domains, it is tagged as an infected machine. Otherwise, it is tagged as clean.

We divided the domain reputation process into two parts; namely, knowledge-based (or domain black and white filtering), and ML-based domain reputation.

1) **DOMAIN BLACK AND WHITELIST FILTERING**

This module describes the formation and updating of domain white and blacklists which implements the knowledge-based domain reputation. The initial blacklist can be taken from threat intelligence providers. In this research we used IBM XForce Exchange [28], Virus Total [29] and BrightCloud [51] API to query the maliciousness of a domain. The initial whitelist was taken from Alexa [27] top 100,000 popular domains. Another whitelist was a monthly domain popularity ranking calculated from the data. The 250 domains that were queried the most, in a month’s time period, were considered benign. Note that updating and searching in lists are much faster procedures than running an ML-based domain reputation. Thus, we used knowledge-based decisions to quickly infer host states with high reliability, while reducing the number of domains we needed to send to the ML-based classifier for deeper inspection.

Note that all hosts must pass through several components in ARBA’s anomaly detection engine before passing through the whitelist filtering. If an infected device was miss-detected it can still be detected in the future based on the domain list it queries or its DNS behavior. It is worth noting that the update
of white lists was designed to accelerate the algorithm over time, and is only optional.

2) MACHINE LEARNING-BASED DOMAIN REPUTATION

This module covers the domain’s feature extraction and Random Forest Classifier (RFC). It inputs a domain name and outputs a score between 0 (benign) and 1 (malicious) that describes its likelihood of being malicious. If the score is above a pre-determined threshold, the domain is considered to be malicious. Otherwise, it is considered to be benign. A classifier is trained on tagged binary domain names based on the passive DNS raw data.

a: DOMAIN FEATURE EXTRACTION

The domain feature extraction module is responsible for extracting meaningful domain features used to classify malicious domains based on passive DNS data. The classifier was trained using monthly aggregated features per domain. In addition, we calculated the number of unique values, the minimum, maximum, mean and standard deviation of some features. The selected features are shown in Table 3. A detailed description of the features is given in the appendix.

| Feature Set | Feature Name |
|-------------|--------------|
| Original | 1. TTL |
| 2. TLD |
| 3. ASN Code |
| 4. Country Iso Code |
| 5. Query Type |
| 6. DNS score |
| 7. Number of unique hosts querying the domain |
| 8. Number of unique resolved IPs |
| 9. Number of different response codes |
| 10. Reputation code type |
| White | 11. Number of days between domain’s creation and expiration |
| 12. Domain age days |
| 13. Domain spamper days |
| 14. Domain in real domain |
| 15. Free mail |
| 16. Number of complaints received |
| 17. Number of person names |
| 18. Registered privacy |
| 19. Email domain score |
| Declaration Probability | 20. Probability of host IP country given TLD |
| 21. Probability of resolved IP country given TLD |
| 22. Probability of resolved IP country given TLD |
| 23. Probability of resolved IP country given TLD |
| 24. Probability of resolved IP country given host IP country |
| 25. Probability of two resolved IPs country to be resolved together |
| Full Query Domain Name (QDNS) string | 26. FQDN and tlds length |
| 27. FQDN length |
| 28. TLD string contains TLD |
| 29. FQDN numerical character percentage |
| 30. TLD upper and lower case number of characters |
| 31. Domain segment name of characters |
| 32. Domain segment name of characters |
| 33. Domain segment name of characters |

b: RANDOM FOREST (RF) CLASSIFIER

The goal of the RF classifier is to distinguish between malicious and benign newly observed domains queried by the hosts marked as suspicious by the anomaly detection engine and the pre-filtering phase. We tested several classification methods, and the RF algorithm achieved the best results. The RF method is an ensemble of decision trees [52]. The basic idea is to combine many decision trees into a single model. Although individual predictions made by one tree may not be accurate, combining the ensemble of trees leads to predictions that have very high accuracy. A subset of the predictor variables was randomly selected to split each node in the tree based on the Gini impurity which is a measure of heterogeneity. The 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. This randomness decreases over-fitting and allows for a higher generalization capability. As a result, it improves the accuracy of the model. Another important advantage of the RF method is that it can measure the importance of each input variable which indicates its contribution to the classification accuracy. Furthermore, RF is not too sensitive to outliers since it is based on decision trees. To compute the Gini impurity for a set of items with \( J \) classes, let \( i \in \{1, 2, \ldots, J\} \), and let \( p_i \) be the fraction of items labeled with class \( i \) in the set. The Gini impurity is given by:

\[
I_G(p) = \sum_{i=1}^{J} p_i \sum_{k \neq i} p_k = \sum_{i=1}^{J} p_i(1 - p_i)
\]

\[
= \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} p_i^2 = 1 - \sum_{i=1}^{J} p_i^2. \quad (6)
\]

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.

D. PSEUDO CODE

Next, we present the pseudo code of the ARBA algorithm. The code describes the algorithm at day \( t + 1 \). For each host in the stable host list, we perform the following. Firstly, 2 lists of domains queried by hosts \( h_t, h_{t+1} \), are extracted. The similarity of these lists for host \( j \), denoted by \( S_{ij} \), is compared to a threshold. hosts that exceed the upper threshold are considered benign, whereas hosts that are below the lower threshold are considered infected. Then, DNS features, \( F_{dj} \), are extracted using the function \( host\_DNS\_Feature\_Extraction \) for the remaining hosts. These features go through the isolation forest anomaly detection algorithm. If the isolation forest fails to acquit the suspected host, features of each new domain of the host, denoted by \( F_{dj} \), are extracted using the function \( Domain\_Feature\_Extraction \) and fed into an RF classifier to label the domain. Finally, if all the host’s domains are benign, it is classified as clean. Otherwise, it is classified as malicious.

IV. EVALUATION AND EXPERIMENTAL RESULTS

We start by describing the labeling method for the dataset used for the algorithm evaluation. Next, we describe the key features of the DRE. Then, we present the algorithm settings and hyper-parameter values used in the experiments and production. We then describe the experiments we tested on real data labeled by cyber security researchers as explained in Section IV-A. Since infected hosts are rare as compared to benign hosts, in real systems, we injected infected hosts to our real data and labeled them accordingly. We compared ARBA performance against the BotDAD algorithm, which recently reported strong performance in infected host detection [26]. Finally, we present real-world scenarios of real station
Here, automatic labeling was based on a combination of two third-party services and white/black lists. The first service was Alexa top 1 Million Global sites list [27] which was used as a white list. The second was Virus Total [29], a service that involves collaboration between leading companies in the cyber security industry, and end users of all kinds, to gather reports about suspicious contents of specific IPs or domain names. The reports contain analyses of URLs and files, user reports, and domain categories according to other third-party services. Once all the information about hostname hourly DNS data is gathered, a statistical analysis is performed.

Because reliable labeling was imperative, to avoid mistakes during the inference procedure, we only used the automatic labeling to label clean hosts, querying benign domains. For each host, the automatic labeling system observes the number of queries per hour and the domains it queries. If it exhibited benign behavior such as querying popular domains and had a normal amount of queries, it was tagged as clean. Otherwise, it was processed manually for verification. Therefore, the hosts shunted to the manual labeling were those labeled by the automatic system with low confidence. We verified these hosts in a 3-stage process. The first stage consisted of gathering information about the domains queried by the suspicious host from XFE [28] and BrightCloud [51]. The second was an analysis of the host, its IP, type of device, activities, and inspections of the domains it queried. The third step involved the manual analysis of this information by a cyber security researcher who drew on the information from the automatic labeling system. To label malicious hosts, we injected the DNS traffic of infected hosts taken from [53]–[55] and from other IBM intrusion detection systems (IDSs). The content of these pcaps was then compared to known blacklists and databases of attacks, and labeled accordingly. At last, if it was not clear whether a host was infected or not, it was not included in the experimental data. This is because we needed data labeled with high confidence for training and evaluation.

### B. FEATURE ANALYSIS FOR THE DOMAIN REPUTATION ENGINE

We performed an extensive feature analysis to design features that could be extracted in real-time, and enable a clear separation between the group of benign domains and the group of malicious domains. One of the most significant and novel features is the probability of communication between the host country and the countries of the resolved IPs of a domain. Let $c_i$ and $r_j$ be the host and resolved IP country, respectively. Let $N_{c_i,r_j}$ be the number of queries from host IPs located in country $i$ to resolved IPs located in country $j$. We defined the probability of communication between host country $i$ and resolved country $j$ by:

$$P_{c_i,r_j} = \frac{N_{c_i,r_j}}{\sum_{i=1}^{M} \sum_{j=1}^{K} N_{c_i,r_j}},$$

where $M$ and $K$ denote the number of unique host and resolved countries in the data, respectively. The feature

---

**Algorithm 1 ARBA Algorithm**

```plaintext
1) for host in host_list do
2) ishostAnomaly = False
3) $l_t = \text{Extract\_Domain\_List}\ (host, \ day_t)$
4) $l_{t+1} = \text{Extract\_Domain\_List}\ (host, \ day_{t+1})$
5) $S_{c_i} = \text{Domain\_List\_Similarity}\ (l_t, \ l_{t+1})$
6) if $S_{c_i} \geq \text{upper\_threshold}$ do
7) ishostAnomaly = False
8) elif $S_{c_i} \leq \text{lower\_threshold}$ do
9) ishostAnomaly = True
10) break
11) else
12) $F_{c_i} = \text{host\_DNS\_Feature\_Extraction}\ (host)$
13) if (Isolation\_Forest($F_{c_i}$)) == 1
14) ishostAnomaly = False
15) else
16) for domain in host\_domain\_list do
17) $F_{d_j} = \text{Domain\_Feature\_Extraction}\ (domain)$
18) $S_{d_j} = \text{Random\_Forest\_Classifier}\ (F_{d_j})$
19) if $S_{d_j} \geq \text{domain\_reputation\_threshold}$
20) ishostAnomaly = True
21) break
22) else
23) domain is benign
24) end if
25) end if
26) end if
27) end if
28) end for
```
distributions of malicious and benign domains in the training data are presented in Fig. 3. As expected, the lower the communication rate, the more likely it is that the host is infected and making queries for malicious purposes such as a malware accessing its Command and Control server.

Another significant feature of the algorithm is the number of days between a domain’s creation and expiration. This feature is based on domain registration information taken from Whois, publicly available data that stores information about registered domains. The feature distribution of malicious and benign domains of the training data is presented in Fig. 4. The longer a domain has been registered, the more likely it is benign. This is because attackers tend to pay for short periods since they expect that the domain will be found and blocked within a short time.

C. PARAMETERS SETTING FOR THE EXPERIMENTAL RESULTS
We set the target recall rate to $R_t = 85\%$ which is common in many cyber security services, and the lower and upper SLD lists intersection thresholds accordingly. We calibrated these thresholds over the training data until satisfying the constraint. A lower threshold of 0.2 and an upper threshold of 0.7 obtained the best results on the validation set. The domain score threshold that achieved the best results on the validation set was 0.35. Then, the IF hyper-parameters were chosen. The IF contains 150 estimators. The maximum depth of each tree was set to $\log_2(n)$, where $n$ is the number of samples. Since we used 8 days of data, which is equivalent to 8 data points, the maximum depth of each tree was 3. Finally, the DRE module’s hyper-parameters were chosen. The RF classifier contains 100 trees. The maximum depth of each tree was bounded to 10 nodes. The optimization of the RF was done using the Gini impurity criteria, as explained in Section III.C.2.

D. NETWORK SIMULATIONS ON REAL DATA
We start by presenting the results using real-data from the IBM network together with simulated data of infected hosts. In these experiments, we compared the ARBA algorithm with the DomainObserver [25] and BotDAD [26] algorithms. To further illustrate the performance of ARBA we present

![FIGURE 3. Histogram of the communication probability between host and resolved countries in benign and malicious domain groups.](image-url)

![FIGURE 4. Histogram of the number of days between the creation and expiration of benign and malicious domain groups.](image-url)
other variations of ARBA that we tested (as detailed below). The data used in the experiments were collected from the IBM network with injections of real infections to the network. The labeling process was performed as explained in detail in Section IV.A. The dataset consisted of nine consecutive days of host DNS queries. These queries were aggregated hourly, per host. In total, 23,760 samples were collected from 110 hosts. These hosts were verified as IoT devices manually. Most of the devices were IP-Telephones and WiFi-based printers. Therefore, all 110 hosts were IoT. Among them, 74 hosts were infected, by injecting malicious domains. The first eight days of data (21,120 samples) were used for training and the ninth day (2,640 samples) was used for testing. The ARBA hyper-parameters were calibrated with the training data. We tuned the parameters of BotDAD and DomainObserver using the same training data.

We start by analyzing the processing time of each algorithm. We include both training time and inference time, although training only needs to be done once or rarely. The running time of ARBA was 11 minutes, on average, to classify a complete hour of all 110 hosts in the data (about 6 seconds per host). Adding this to the training time of the domain reputation classifier, which took approximately 12 minutes, results in an average of 23 minutes to classify all the hosts in the data. Thus, the system meets Constraint C2 which requires that the total processing time should be less than one hour. In terms of memory usage, the measurements showed that ARBA used up to 200MB memory. The average memory usage measured is 160MB. As for scalability, the complexity meets the constraint of $O(N)$, where $N$ represents the number of domains. Moreover, ARBA scalability was demonstrated daily by operating in a production environment in a large-scale real-time IBM network.

Table 4 presents a detailed comparison of the following algorithms: (i) The proposed ARBA algorithm; (ii) ARBA with SVM instead of RF classification in the domain reputation inspection phase, dubbed ARBA-SVM; (iii) ARBA with logistic regression, dubbed ARBA-LR; (iv) ARBA without a classifier, dubbed ARBA-NC; (v) the BotDAD algorithm [26]; and (vi) the DomainObserver algorithm [25]. The bold-faced row presents the results obtained by the suggested ARBA algorithm. The first column shows the tested algorithm, and the second, and third columns present the precision and recall rates, respectively. The fourth column shows the accuracy: $A = (TP + TN)/(TP + FP + TN + FN)$, and the fifth column shows the $F_1$ score: $F_1 = 2 \cdot P \cdot R/(P + R)$.

The last column presents the inference time; i.e., the time it took to classify a host on average in seconds. As can be seen from Table 4, the proposed ARBA algorithm achieved very strong performance. The recall met the target recall $R$, and the resulting precision was higher than the precision achieved in BotDAD and the other tested algorithms. In addition, ARBA outperformed all other algorithms under each measured metric (except the prediction time as discussed next). Finally, we tested the prediction time per host; i.e., the time it takes for the algorithm to label the host given all the data. It can be seen that ARBA was slightly faster than BotDAD in this respect as well, although slightly slower than ARBA-SVM and ARBA-LR. DomainObserver was the slowest. ARBA-NC was the fastest as expected since it does not use a classifier. In the experiments, 74 out of 2640 test samples (110 hosts *24 hours) were infected. Out of 74 test samples, 69 hosts were classified infected (correctly), 2 hosts were false positives, and 5 samples were false negatives. For each detected infected host (total 69), we alert once to avoid filling the infected hosts database with the same hosts repeatedly.

### TABLE 4. Performance evaluation.

| Algorithm    | $P(\%)$ | $R(\%)$ | $A(\%)$ | $F_1(\%)$ | $T(s)$ |
|--------------|---------|---------|---------|-----------|-------|
| ARBA         | 97.1    | 93.2    | 99.7    | 95.1      | 6.0   |
| ARBA-SVM     | 97.1    | 90.5    | 99.6    | 93.6      | 5.7   |
| ARBA-LR      | 93.0    | 90.5    | 99.5    | 91.8      | 5.3   |
| ARBA-NC      | 92.9    | 89.1    | 99.5    | 90.9      | 3.7   |
| BotDAD       | 94.2    | 87.8    | 99.5    | 90.9      | 6.4   |
| DomainObserver | 87.1   | 82.4    | 99.1    | 84.7      | 28.9  |
anomalies detected by ARBA. We present two scenarios of infected hosts and the domains that the algorithm used to frame them.

In Figure 5 we present an example of a host, assessed by the anomaly detection as infected. The device is an IoT of a family of printers or IP-telephones. The family is characterized by a static behavior. The DNS pattern of such behavior is querying the same 1-3 domains periodically. The blue dots and red dots in the figure represent the training and test data, respectively. The test sample is marked in red and indicates an anomaly. Although there are eight samples of training data, some of the blue dots are hidden by other dots, resulting in only five visible points.

![Figure 5](image1.png)

**FIGURE 5.** The number of queries versus the number of unique SLDs queries for infected and clean sample features.

Our analysis showed that for this scenario, the two most significant features of the IF output were the number of queries and the number of unique SLDs queried by the host that day. In the training data, the host made 6 queries on average per day, mostly to one unique SLD. In the test data on the other hand, the host made 13 queries to 4 unique SLDs. As expected, the algorithm marked the test sample as an anomaly and passed it to the DRE. The domains were compared to black and white lists. Three out of the four unique SLDs queried by the host were in the Alexa top 1-Million domains and were labeled benign domains. The last domain:

```
iuqer f sodpfj japosdfj hgosuri j faewrwegw f f .com
```

did not appear in the black or white lists and was sent to the RF classifier for further analysis. The RF classified the domain as malicious, thus determining that the host was infected. After making an in-depth analysis, we confirmed that the detected domain is indeed malicious. Specifically, in [56], the authors performed a dynamic analysis of WannaCry ransomware, and showed that the domain was an indicator of compromise for WannaCry ransomware. Other security researchers such as Benkow and Matt Suiche confirmed this claim. These previous reports, and the high probability score of being malicious from XFE and Virus Total [28], [29] support our conclusion.

In the second scenario we tested, we could not detect two main features that separated the anomaly from the rest of the data. Therefore, to distinguish between them, we performed dimensional reduction using PCA as suggested in [57]. The results are presented in Figure 6. It can be seen that the test sample was classified as an anomaly. Then, when analyzing each domain queried by the host, the Domain DRE classified one of the domains, disorderstatus.ru, as malicious. After analyzing the domain we discovered that the website contained malware categorized as Adware; thus the domain was malicious. In addition, the malware directed the user to fraudulent sites. The malware was still active and communicating with malicious files according to Virus Total [29]. This can be seen in Figure 7, where the “last seen” of the malicious files are up to date (Nov. 2019). The score highlighted in the image is the maliciousness score from Virus Total engine, where zero means benign. An additional analysis was done using the Virus Total Graph [29], as can be seen in Figure 8. Malicious files are marked in red, and clean files in black. It can be seen that disorderstatus.ru communicates with several malicious files. Furthermore, there are malicious files for download within the domain, and some of its URLs are malicious as well.

![Figure 6](image2.png)

**FIGURE 6.** The 3D representation of 3 Components PCA.

![Figure 7](image3.png)

**FIGURE 7.** An illustration from Virus Total analysis for the disorderstatus.ru domain that we identified in our experiments. Files that communicate with disorderstatus.ru were classified as malicious.

These scenarios demonstrate the effectiveness of ARBA in a real-time production environment in large-scale networks.
V. CONCLUSION

In this paper we developed a novel Anomaly and Reputation Based Algorithm (ARBA) for detecting infected IoT hosts via DNS passive data inspection in large-scale networks. ARBA is a computationally efficient, light weight and scalable real-time algorithm. It has two main modules. The first module performs host infection status analysis by comparing domains queried by the host in the past and present. Then, anomaly detection for the host is implemented using isolation forest. The second module classifies malicious domains queried by each suspected host. This phase is done using both black and white lists followed by feature extraction whose output is processed through a trained random forest classifier. We performed extensive network simulations and compared ARBA with BotDAD which reported strong results recently in detecting infected hosts using hourly aggregated features. We showed that ARBA outperformed BotDAD on all the tested measures, particularly in precision, recall and prediction time.

We further developed ARBA on a real-time production environment. We demonstrated that ARBA works well on the production environment, and detected existing malicious activities over DNS in real-time. We chose to use Alexa in our development since only 0.057% of Alexa domains were flagged as malicious (as compared 0.22% of Majestic Million) based on the malware existence analysis in top-sites ranking in [58]. A potential research direction is to examine Majestic Million for future development.

Finally, although ARBA aims to detect infected IoT hosts, we expect that it can be found useful in detecting other infected hosts that query a limited number of domains daily.

Our research group is currently exploring the possibility of applying the system to applications running on Kubernetes environments, since it is assumed that containerized services mostly query the same domains daily.

APPENDIX A

DESCRIPTION OF THE HOST’S FEATURES

In this appendix we detail the specifics of the features in Table 2.

Feature (F1): number of unique hours of queries: This feature gives a strong indication whether a host is infected by looking at the daily queries over time distribution. A motivation can be derived by the following use case. In a clean environment, a printer communicates with its server every 12 hours approximately, leading to F1 value of 1 or 2. In a case of an infection such as in Mirai botnet, the printer would query randomly every few of hours, leading to F1 value of more than 5 [50]. The value range of this feature is between 0 and 24.

F2/F3: Number of daytime/nighttime queries: IoT hosts tend to query their control server periodically for a firmware update [37]. Moreover, many hosts query their server at a predetermined time. Botnets, on the other hand, tend to query their server irregularly during a period of 24 hours. The value range of these features is between 0 and 12.

F4/F5: Number of max queries in 1 minute/1 hour window: These features aim to detect DDoS attacks. In addition, the time interval between queries of a botnet is much smaller than the interval of a clean host. Burst of queries is a symptom of botnet infection.

F6: Number of unique query types: Most IoT hosts query the same type of queries and therefore their F6 value typically equals 1. However, in a case of infection it is probable that an IoT host would communicate with its C&C using other types of records. For example, it is unlikely that a benign printer, that usually queries only A types of queries, making its F6 value equals 1, would suddenly use Mail exchanger record (MX Record), which results in F6 value of 2.

F7: Query type: F7 is a counter vector that counts the number of queries from each query type.

F8/F9: Number of unique response codes/Response code type: These features mirror features F6/F7 for the response codes.

F10: Number of unique domains queried: IoT hosts usually query the same domains. Infected hosts, on the other hand, query a wider range of domains.

F11: Number of Alexa domains queried: This feature counts the number of domains queried, which are listed in Alexa. Clean IoT hosts mostly query benign domains. According to [58], less than 0.15% of Alexa top 100K are malicious domains. Moreover, according to extensive experimental analysis that we have made, we obtained that an infected host has a higher probability of querying non-Alexa domains.

F12-F15: Minimum queries per domain/maximum queries per domain/mean of queries per domain/standard deviation
of queries per domain: These features consist of statistical measures of queries (minimum, maximum, mean, std), aggregated per domain. A sharp increase or decrease in any of these values indicates anomalous for IoT hosts or any other static DNS behavioral host.

**APPENDIX B**

**DESCRIPTION OF THE DOMAIN’S FEATURES**

In this appendix we detail the specifics of features in Table 3.

F1: TTL: This feature stands for domain time to live.

F2: TLD: This feature stands for top level domain. This feature is extracted from the full query domain name (FQDN).

F3/F4: ASN code/country ISO code: These attributes consist of the location of domain name registration (autonomous system and country).

F5: Query type: This feature is a counter vector that counts the number of queries from each query type.

F6: XFE score: This feature is the domain reputation historical score taken from IBM XFE [28].

F7/F8/F9: Number of unique hosts querying the domain/number of unique resolved IPs/number of different response codes: This feature counts the number of unique hosts querying the domain, unique resolved IPs, and number of different response codes, respectively.

F10: Response code type: This feature is a counter vector that counts the number of response code from each response code type.

F11: Number of days between domain’s creation and expiration: This feature highly correlates with the domain maliciousness. A domain with small F11 value is most likely to be malicious while a domain with a large F11 value is likely to be benign.

F12: Domain age days: This feature counts the number of days since the domain was created. We observed that the higher this value, the more likely the domain is benign.

F13: Domain update days: This feature counts the number of days since the domain last registration update.

F14: Domain to mail distance: This feature computes the Jaro distance between the domain name and the domain name which was extracted from the WHOIS record’s contact email.

F15: Free mail: This is a Boolean feature, indicating whether the registrant pays for mail or uses free mail services. Paid mail is more likely to be benign.

F16: Number of person names: This feature counts the number of person names which appear in one of the contact names from the WHOIS records.

F17: Number of corporation names: This feature counts the number of corporation names which appear in one of the contact names from the WHOIS records.

F18: Registration privacy: This is a Boolean feature, equals TRUE if the domain was registered using domain privacy registration services, or FALSE otherwise. Privacy registration service is identified if the email address or one of the contact names from the WHOIS record contains one of the words: 'privatemailprivacy|protectwhoisproxy'

F19: Email domain variety: This feature computes the ratio between the number of emails which are provided in the WHOIS records and the number of unique domains for these emails.

F20-F25: Probability of host IP country given TLD/probability of resolved IP country given TLD/probability of resolved IP country given TLD/probability of host IP country given resolved IP country/probability of resolved IP country given host IP country/probability of two resolved IPs countries to be resolved together: These features compute the corresponding empirical probabilities by counting the number of event occurrences (e.g., communication between specific host IP country and specific TLD in F20) over the total DNS data (e.g., communication between this specific host IP country and all possible TLDs in F20).

Next, for convenience, assume that given FQDN “rosenthal.ibm.com” there are 2 tokens, namely “rosenthal” and “ibm”. We will describe features F26-F33 below using this example.

F26: FQDN max token length: This feature counts the number of characters in the longest token. In the example, “rosenthal” is the longest token, containing 9 characters.

F27: FQDN length: This feature counts the length of the FQDN string. In the example, the length of “rosenthal.ibm.com” equals 17.

F28: SLD string contains TLD: This is a Boolean feature, indicating whether the subdomain contains a top-level domain. In the example, “.com” is a TLD, therefore the value is True.

F29: Numerical characters percentage: This feature computes the ratio between numerical characters and other characters. It is computed by dividing the number of numerical characters by the total number of characters (excluding the dots) in the FQDN. In the example, this value equals 0 since there are no numerical characters in the given FQDN. For a different FQDN, e.g., “rosenthal.ibm.com” we get an F29 value of 1/16 = 0.0625.

F30-F33: Token minimum number of characters/token maximum number of characters/token mean number of characters/token standard deviation number of characters: These features are computed similarly to F29. However, instead of FQDN, the computation is done aggregately per token. For example, in “rosenthal.ibm2m.co4mm”, F30 = min(1/10, 1/4, 1/5) = 1/10.

**REFERENCES**

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[2] A. L. V. Caraguaya, A. B. Peral, L. I. B. López, and L. J. G. Villalba, “SDN: Evolution and opportunities in the development IoT applications,” *Int. J. Distrib. Sensor Netw.*, vol. 10, no. 5, May 2014, Art. no. 735142, doi: 10.1155/2014/735142.

[3] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, “A survey on IoT security: Application areas, security threats, and solution architectures,” *IEEE Access*, vol. 7, pp. 82721–82743, 2019.

[4] Y. Zhauniarovich, I. Khalil, T. Yu, and M. Dacier, “A survey on malicious domains detection through DNS data analysis,” *ACM Comput. Surv. (CSUR)*, vol. 51, no. 4, p. 67, 2018.
[55] Internet Systems Consortium (ISC). ISC Malicious Domains. Accessed: Jan. 2020. [Online]. Available: https://isc.sans.edu/suspicious_domains.html

[56] D.-Y. Kao and S.-C. Hsiao, “The dynamic analysis of WannaCry ransomware,” in Proc. 20th Int. Conf. Adv. Commun. Technol. (ICACT), Feb. 2018, pp. 159–166.

[57] K. Pearson, “LIII. On lines and planes of closest fit to systems of points in space,” London, Edinburgh, Dublin Phil. Mag. J. Sci., vol. 2, no. 11, pp. 559–572, Nov. 1901.

[58] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen, “Tranco: A research-oriented top sites rank hard-ened against manipulation,” 2018, arXiv:1806.01156. [Online]. Available: http://arxiv.org/abs/1806.01156

GILAD ROSENTHAL received the B.Sc. degree in electrical engineering from the Ben-Gurion University of the Negev, 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 was a Data Scientist with start-ups and corporates. His current research interests include machine learning, artificial intelligence, deep learning, anomaly detection, and data science in cybersecurity domain.

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

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

AVISHAY BARTIK received the B.Sc. degree in mathematics and computer science from The Open University of Israel, in 2010. He was a Security Software Engineer with PMO. He is currently a Security Researcher with IBM Cyber Security Center of Excellence, Beer-Sheva, working on various aspects of network and system security.

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

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