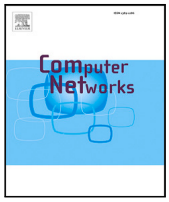
Computer Networks 228 (2023) 109725

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Real-time bot infection detection system using DNS fingerprinting and machine-learning

Vicente Quezada [a], Fabian Astudillo-Salinas [a,*], Luis Tello-Oquendo [b], Paul Bernal [c]

[a] DEET, Universidad de Cuenca, Ecuador
[b] College of Engineering, Universidad Nacional de Chimborazo, Ecuador
[c] CEDIA corporation, Ecuador

ARTICLE INFO

Keywords:
Botnet
Bot detection
DNS-based bot detection
Anomaly detection
ELK stack
Machine learning
Isolation forests
Random forests

ABSTRACT

In today's cyberattacks, botnets are used as an advanced technique to generate sophisticated and coordinated attacks. Infected systems connect to a command and control (C&C) server to receive commands and attack. Thus, detecting infected hosts makes it possible to protect the network's resources and prevent them from illicit activities toward third parties. This research elaborates on the design, implementation, and results of a bot infection detection system based on Domain Name System (DNS) traffic events for a network corporation. An infection detection feasibility analysis is performed by creating fingerprints. The traces are generated from a numerical analysis of 13 attributes. These attributes are obtained from the DNS logs of a DNS server. It looks for fingerprint anomalies using Isolation Forest to label a host as infected or not. In addition, on the traces cataloged as anomalous, a search will be carried out for queries to domains generated by Domain Generation Algorithms (DGA). Then, Random Forest generates a model that detects future bot infections on hosts. The devised system integrates the ELK stack and Python. This integration facilitates the management, transformation, and storage of events, generation of fingerprints, machine learning application, and analysis of fingerprint classification results with a precision greater than 99%.

1. Introduction

Current attacks such as spam, Distributed Denial-of-Service (DDoS), click fraud, malvertising/ad fraud, identity theft, and corporate espionage have as common source Botnets [1]. Botnets are groups of infected hosts (also known as bots) controlled by an attacker (botmaster). Generally, users are unaware that a hidden backdoor on their host is being used for criminal purposes, thus becoming the most significant threat on the Internet.

Compromised machines establish a command and control (C&C) channel from which malicious activities can be executed not only towards the infected host but used to attack other hosts on the network. In this context, different C&C architectures and communication techniques have been developed by attackers to make any channel detection and disruption process difficult. One of these techniques is using the Domain Name System (DNS) service to resolve domain names associated with a C&C server. Therefore, detecting and blocking communications with malicious domains will disrupt secret channels between victims and controllers, thus limiting harmful effects on network resources [2]. Although significant efforts are made to detect botnets globally, little is made to detect bot infections at the enterprise level [3,4].

Various network-based botnet detection techniques that are based on how the traffic network behaves have been proposed [5]. A DNS rule-based approach for Botnet detection (DNS-BD) is presented in [6], which can increase the precision of DNS traffic-based botnet identification. By using the suggested DNS query and response rules, this method based on DNS query and response behaviors, seeks to identify any unusual DNS query and answer behaviors. Based on the suggested flow analysis methodology, the authors of [7] classified network traffic behavior and discovered botnet activity using a decision tree with the reduced error pruning algorithm (REP Tree); the suggested classification approach accurately detects the activities of known and unidentified botnets even within a very narrow time frame. The authors in [8] were able to recognize the aberrant traffic across a sizable amount of typical network data during the communication stage with the botnet server without prior knowledge of the botnets or servers. Using similarity measurement and periodic botnet features, anomaly score-based botnet detection is proposed in [8] as a method of detecting botnet activities. To identify domain-flux botnets, DFBotKiller was devised in [9], a negative reputation system that takes into account

* Corresponding author.
E-mail addresses: vicente.quezada@ucuenca.edu.ec (V. Quezada), fabian.astudillos@ucuenca.edu.ec (F. Astudillo-Salinas), luis.tello@unach.edu.ec (L. Tello-Oquendo), paul.bernal@cedia.org.ec (P. Bernal).

https://doi.org/10.1016/j.comnet.2023.109725
Received 3 October 2022; Received in revised form 16 February 2023; Accepted 16 March 2023
Available online 23 March 2023

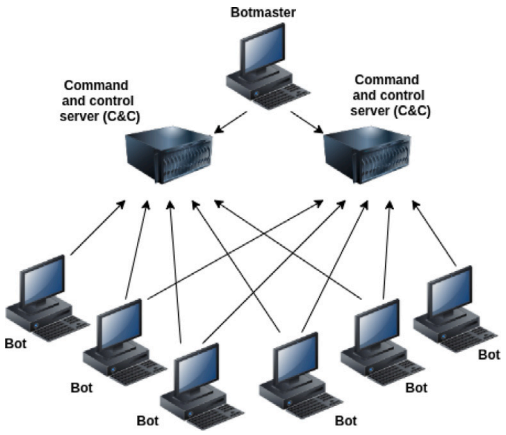


Fig. 1. Botnet architecture.

the history of both suspicious group actions and suspicious DNS traffic failures.

Detecting bot-infected machines is vital for any organization to combat security threats and take appropriate corrective actions. To solve these issues, several studies propose different solutions and architectures that capture network traffic and can work offline [10] and online [3]. The authors of the last one proposed a new work based on this. The authors called their framework DeepDAD [4]. From our experience, these solutions are difficult to integrate into a corporate network because, first, the security manager has to modify the policies to incorporate a port mirroring; second, it is technically difficult to enable this port mirroring for some corporate network architectures. Furthermore, most existing solutions do not share source codes for reproducibility purposes. The exception is DeepDAD [4]; the authors of this work shared their source code in the github repository https://github.com/mannirulz/DeepDAD. Based on these observations, we propose a solution that works online and uses real-time DNS logs to detect botnets and feed these data into incident management systems and provide alerts to security analysts. Also, we made our source codes and a docker-compose available for quick integration into network infrastructures.

To develop this study, we consider the data and infrastructure of a corporate network named CEDIA. It is an internet service provider for most of the higher education institutes in Ecuador. Its mission is "To work collaboratively to generate and strengthen research networks at the national and international level, for the benefit of society". We used CEDIA's cloud services to implement the proposed real-time bot infection detection system described in further sections.

1.1. Preliminaries

1.1.1. Botnet architecture

The typical architecture of a Botnet is presented in Fig. 1 and consists of the following components:

- **Bots:** They are devices connected to the Internet that are infected with malware and are controlled remotely.
- **C&C Server:** It is a server that controls bots on the network through broadcast commands.
- ***Botmaster:*** It is a person who takes control of the C&C servers and sends commands to bots of a Botnet.

1.1.2. Life-cycle of a bot-infected host

The behavior of an infected host is analyzed in [11]; a general pattern is established that can be described in four phases as detailed in the following.

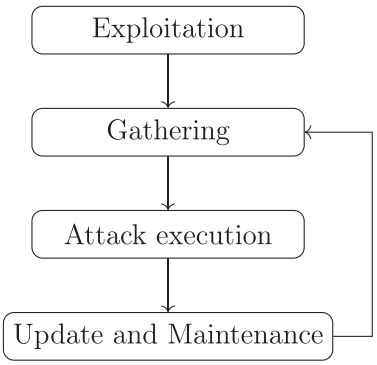


Fig. 2. Life cycle of a botnet [11].

- **Exploitation phase:** The botmaster infects a victim host by exploiting an existing software vulnerability. The infection goes hand in hand with some fraud towards the victim to execute a malicious code on his machine. The connection to the remote server is established only after the compromised machine issues a DNS lookup command, seeking to assign an IP address to its remote server.
- **Gathering phase:** The bots continuously connect to their botmaster via a C&C server. The botmaster equips its bots with a DNS lookup functionality to make queries and locate the C&C server. IPs constantly migrate to prevent server IPs from being included in security blacklists, thus hiding their address behind a domain name. As a result, the bots will continue to communicate with the botmaster. This communication can occur once the DNS request to the C&C server is resolved.
- **Attack execution phase:** The bot group performs malicious activities on the target machines by receiving commands from the C&C servers provided by the botmaster.
- **Update and maintenance phase:** The botmaster instructs the bots to update their binaries from time to time for better coordination. Furthermore, botmasters must frequently migrate their C&C server location to evade various bot detection techniques.

The flow of these phases in the life cycle is presented in Fig. 2. The DNS traffic analysis on this life-cycle has the potential as a technique to detect botnets [11]. Thus, this technique is considered a promising field of research to combat botnet threats.

1.1.3. Botnets and DNS

As mentioned before, the life cycle of a botnet depends mainly on the domain name resolution of the C&C servers. Attackers or botmasters use domain name services to hide their C&C IP addresses so that the botnet is reliable and easy to migrate from server to server without raising suspicions. Thus, botnet detection techniques based on DNS traffic analysis are the most used in their detection [11]; among these techniques, Domain Generation Algorithms (DGA)-based detection and bot-infected hosts detection are the most widely used nowadays.

- **DGA-based detection:** It attempts to differentiate the domains queried in a network that are algorithmically generated (malignant) from regular domains. DGA-generated domains are pseudo-random names because they are incomprehensible to humans. Domains do not form phrases that can make sense of a real address.
- **Bot-infected hosts detection:** It attempts to detect bot-infected machines on a network instead of finding the C&C server. The main focus of these techniques is to list the infected hosts on a network.

1.2. Contribution

We build a real-time system that allows detecting infected hosts in an active corporate network. The software tools and procedures to achieve this objective are presented. It focuses on detecting bot-infected hosts, using hourly DNS fingerprinting of active hosts on the network. Unsupervised learning is applied to these tracks focused on anomaly detection. In addition, we implement a detection technique using a randomness measurement algorithm for a DGA-based domain with a whitelist; the DGA-based algorithm is based on [10,12,13]. Finally, supervised learning seeks to generate a model to detect possible bot infections on future footprints.

For reproducibility, it is worth mentioning that the algorithms and procedures devised were implemented on the OS Ubuntu 20.04.1 LTS. These algorithms can be found on the collaborative development platform GitHub in the following link https://github.com/fabianastudillo/bndf.

The main contributions of the study can be summarized as follows:

- A novel architecture for real-time bot infection detection is devised. It exploits data from log servers and uses DNS fingerprinting and machine learning for effective detection. Furthermore, it can be deployed quickly.
- Network event generation, digestion, and analysis tools based on the ELK stack for network intrusion detection are developed and provided for reproducibility.
- Thirteen attributes from DNS logs were identified.
- Anomalies' detection related to bot infections is elaborated considering fingerprints generated from DNS log events.
- Implement an algorithm based on machine learning to detect DGA-based domains in an active corporate network.

The rest of the article is organized as follows: a literature review focused on botnet detection based on DNS flow analysis is carried out in Section 2. The architecture and techniques used for the project are presented in Section 3. The implementation, evaluation, and results analysis are presented in Section 4. Finally, the conclusions and future works are covered in Section 5.

2. Related work

There are several botnet detection mechanisms proposed in the literature. In the review article [14], the authors summarize the techniques. They divide the technologies of botnet detection into three categories based on honeypot analysis, communication signatures, and abnormal behavior. Instead, the authors of [15] divide into honeynet and intrusion-based detection techniques; inside the last one are included the signature and the anomaly-based detection techniques. Our work is into the anomaly-based detection technique. According to the review article [14], the anomaly-based detection techniques are: deep learning, complex networks, swarm intelligence, statistical analysis, distributed approach, and finally, combination method. Although not explicitly mentioned, machine learning is also found.

Regardless of the method, input data is needed to feed the botnet detection methods; the techniques use network traffic for example, the authors of [16,17]. But using this kind of traffic several features have to be extracted and the more traffic, the more processing. Another reason is that DNS request is usually the first step to contact the C&C server of the bots; these are controlled by the bot master, and the detection of the DNS request is effective in detecting the bots. Actually, most of the state-of-the-art use DNS traffic as input data.

Several DNS-based botnet detection techniques are examined in [11]. This study explores solutions and directs future research toward passive botnet detection techniques based on traffic analysis of DNS to achieve adequate detection mechanisms. In this context, [2] presents the state-of-the-art of problems and challenges of detecting botnets based on DNS. Consequently, a new classification for DNS-based botnet detection techniques is introduced and categorized as flow-based detection, anomaly-based detection, DGA communication detection, and bot-infected hosts detection on a network. Emphasis is placed on the last two methods as new research approaches. With this previous classification, some studies are presented below.

In [18], the authors present techniques for botnet detection based on analyzing the flow and anomalies of the DNS traffic. First, 23 characteristics are extracted from the net DNS flow. Later, the legitimacy of the domain name is determined using machine learning and thus detecting botnets. From the applied machine learning algorithms, Random Forest achieved an accuracy (ratio between the correct predictions obtained and those predicted) of 99%.

Most botnets based on DNS protocol adopt DGA; these botnets can change the domain randomly to hide themselves. A tool that takes advantage of the behavior of a DGA communication named DBod is presented in [1]. It is assumed that out of multiple domains queried, only a few domains are associated with an active C&C. Therefore, the job looks for NXDomain or failed query surge anomalies using clustering techniques.

In [19], a method to detect DGA domains from massive DNS queries is presented. It uses six dictionaries: English, French, German, Spanish, Russian, and Japanese. Based on these dictionaries, the randomness of the domains is estimated through a lexical analysis, achieving a precision of 90%. Other studies have also been developed in detecting DGA-based domains, such as [13]; this work was improved in [13]. The study considers a language-independent deterministic algorithm. It seeks to detect incomprehensible domain names based on general sonic axioms, such as the number of consequent vowels, the number of consequent consonants, and the domain's entropy. Depending on the analyzed family, the algorithm achieves precision values above 90%. The method is proposed as an efficient first alarm for recognizing botnet communications.

Other authors also use the DGA techniques. In [20], the authors present BotHook; it is a supervised machine learning approach for botnet detection using DNS query data. They conclude that Support Vector Machine (SVM) gives the best outcome among other machine learning techniques; the tool used by the authors is WEKA. The authors of [21,22] propose a botnet detection architecture based on an artificial neural network and machine learning, respectively.

A methodology for detecting host anomaly behavior (i.e., for detecting hosts infected by bots) is presented in [23]. It looks for chains of attack patterns or Kill Chain to detect C&C channels, pharming, and IP spoofing DDoS botnets. The scheme creates behavior profiles for each host on the network. However, it requires a large amount of storage and computational capacity since all network traffic is considered. In [24], the term DNS fingerprint is introduced as a host's DNS communication behavior profile. Each behavioral fingerprint uniquely identifies its corresponding user and is immune to time change. This way, a user has an invariable DNS behavior pattern on the network. Furthermore, with the generated fingerprints of the hosts, it is sought to detect new flows that would become abnormal behaviors in the network.

BotDAD [3] is introduced as a new anomaly-based detection technique, which considers the DNS fingerprint of hosts per hour and tries to find anomalous behavior of a machine. First, fingerprints are generated by extracting 15 characteristics from the DNS flow for each host. Then, a detection engine based on an anomaly search is made with the fingerprints generated. Finally, a Random Forest machine learning algorithm is trained to classify clean and infected hosts in future captures. A continuation of the work done in [3] is presented in [25]. The machine-learning algorithm of the BotDAD anomaly detection engine is improved for future fingerprints. Thus, a new tool called DeepDAD is created using multilayer neural networks [4]; in this work, the authors conclude that multi-feature anomaly detection is a better measure of detecting malicious hosts in a network, and that deep neural networks have slight yet significant improvements compared to random forest.

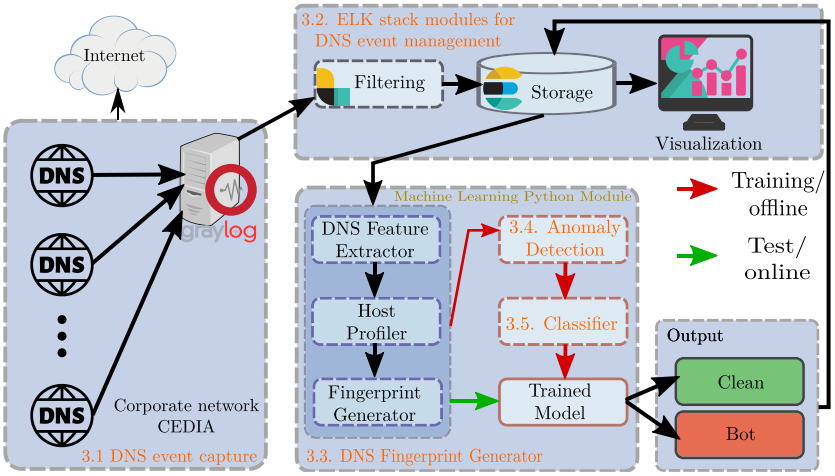


Fig. 3. Architecture of the solution from the processes perspective.

The authors of [26] propose a botnet detection method by using a hybrid of particle swarm optimization (PSO) algorithm with a voting system. The PSO algorithm is employed to select features, and the voting system is used to identify botnets and classify samples. According to the authors, the method improves the accuracy by an average of 0.42% and 0.17% in the ISOT data set and the Bot-IoT data set, respectively, compared to the other methods investigated.

The works using the DNS query to feature extraction are [27–30]. The authors of [27] propose an optimized ML-based framework to detect botnets based on their corresponding DNS queries; the framework consists of using information gain as a feature selection method and genetic algorithm (GA) as a hyperparameter optimization model to tune the parameters of a random forest (RF) classifier. The approach of authors of [28] is to process the timing information of the generated DNS queries to identify the existence of the group activity behavior (i.e., bots behavior) by measuring the level of similarity between periodic hosts. The paper [29] examines the abnormality of DNS traffic during the botnet lifecycle to extract significant enriched features. These features are further analyzed using two machine learning algorithms. The union of the output of two algorithms proposes a novel hybrid rule detection model approach. In recent years, Botnet attacks towards the Internet of Things (IoT) have been considered to be the attacks with the most extensive impact on Internet infrastructure; the authors of [30] designed an IoT device detection systems to detect the devices infected by botnets using the traffic based on DNS.

3. Bot detection architecture based on machine learning

The generic architecture of a network anomaly detection system is composed of four main modules, namely:

– *Traffic capture:* In this module, live network traffic is captured using some sniffer;
– *Information processing:* This module filters the relevant parameters during capture and feature extraction from the captured data; it also takes care of the conversion, normalization, and discretization of the filtered data type into information in a format readable by other modules;
– *Anomaly detection:* This module is the heart of any network anomaly detection system; it attempts to detect the occurrence of any intrusion, whether online or offline; if the attack belongs to a known type, it can be detected using a misuse detection approach. Unknown attacks can be detected with the anomaly-based approach using an appropriate matching mechanism or classifier;
– *Notification:* This module is responsible for generating an alarm based on the indication received from the anomaly detection engine. In addition to indicating the occurrence of an attack, alarms are helpful for further analysis. For example, alarms should indicate any background information that justifies the generated alarm.

Additionally, an essential element is a *human analyst*, responsible for analyzing, interpreting, and taking necessary actions based on the alarm information provided by the detection engine. The analyst also follows the necessary steps to diagnose the alarm information as a post-processing activity to support the update of the reference or profile.

Based on the model mentioned above, we proposed a novel architecture for a real-time bot infection detection system that includes event capture, event management, fingerprint generation, and machine learning for the anomaly detection modules. This architecture is illustrated in Fig. 3. The modules that intervene in the data processing, generation of DNS fingerprints, and anomaly detection are presented with their respective processes to obtain the desired output. The system is based on the DNS log capture. DNS logs are sent to a log server, and this forwards the logs to the BotNet Detection Framework (BNDF). The BNDF performs the filtering storage, anomaly detection using the logs, and visualization. The system acts offline in a steady state. This state takes ten days to analyze all the DNS traffic and generates a trained model using random forest. After this stage, the system can be run in online mode. In this mode, the calculated fingerprints are sent to the trained model and marked as bot or clean. We could not compare results with related works since the source codes of the algorithms are not shared for reproducibility purposes. Furthermore, the related works operate offline, and the detection is not dynamic, which makes the comparison unfeasible. Note that this work aims to be the baseline so that modules can be added to our framework in the future (the code is available on github platform) and the algorithms can be compared effectively. The different implemented algorithms were the best ones in the state-of-the-art after analyzing the discussion and results provided.

In the following, we elaborate on the main features of each architecture component.

3.1. DNS event capture

The architecture has been designed to use several DNS servers. All these servers send their logs to a Graylog server. This Graylog server in CEDIA forwards the logs to the BNDF virtual machine implementing four dockers: Logstash, Elasticsearch, Kibana, and Python (machine learning module). A brief description of each tool follows:

Table 1
Relevant fields of DNS events stored in Elasticsearch.

Field	Description	Types
dest_ip	destination IP address	various
dn.sld	second level mastery	various
dn.tld	top-level domain	various
dns.rcode	return code	NOERROR NXDOMAIN
dns.rrname	domain name consulted	various
dns.rrtype	type of resource record requested	A AAAA PTR MX
dns.type	Event type	query answer
src_ip	source IP address	various

Graylog: It is a centralized Log Management System (LMS) that provides a means to aggregate, organize, and make sense of all this data [31].

Logstash: It is a free and open server-side data processing pipeline that ingests data from many sources, transforms it, and then sends it to a "stash" [32].

Elasticsearch: It is a distributed, RESTful search and analytics engine capable of addressing many use cases. It centrally stores the data for lightning-fast search, fine-tuned relevancy, and powerful analytics that scale easily [32].

Kibana: It is a free and open user interface that lets us visualize the Elasticsearch data [32].

Python: It is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming [33].

When used generically, the term encompasses a larger system of log collection, processing, storage, and searching activities. In this case, the input module listens to a specific port by Graylog Extended Log Format (GELF).

This architecture has been proposed because in the information and communications technology (ICT) department of the corporate network CEDIA a Graylog server that stores DNS event logs is implemented.

3.2. ELK stack modules for DNS event management

The ELK stack will manage the GELF events generated by Graylog. Logstash will extract top-level and second-level domains from the domain name (TLD and SLD, respectively). Elasticsearch, in turn, serves as a storage base and search engine for future queries. Finally, Kibana is the visual interface to interact with the data and query lab. Projects such as SELKS [34], and those described in [35,36] demonstrate the correct operation of using Elasticsearch, Logstash, Kibana, and Suricata as a set; in our case, we use the first three tools and replace Suricata by the DNS log server.

The records obtained by processing the events generated by Graylog with Logstash have the characteristics detailed in Table 1 referring to a log type answer, where several fields are typical of Elasticsearch to be able to index and manage the stored events.

3.3. DNS fingerprint generation

The detailed methodology in [3] is considered to develop this architecture component. It consists of three modules: (1) DNS Feature Extractor, (2) Host Profiler, and (3) Fingerprint Generator. The *DNS feature extractor module* processes the DNS logs (e.g., number of transactions, Host IP, FQDN, number of domain tokens, request type, length of FQDN, request timestamp, and the DNS server IP) and response parameters (e.g., transaction number, Host IP, FQDN, query type, answer code, TTL, resolved IP, and response timestamp) that are useful for fingerprint generation. The output of the DNS extraction module, or the request and response parameters, are parsed by the *host profiler* module. Organizing the requests from each host into a schema creates a DNS profile for every host on the network. Each host's DNS request is further categorized using FQDN. A host can make several requests for the same FQDN by using a list. The typical DNS query reply message has many answer records. A list is kept up to date to manage these answer records for a specific DNS query request. An in-memory profile of all the hosts that used the DNS service is obtained when the processing is complete. Then, the *DNS fingerprint generator* module parses the in-memory host profile, creating a unique DNS fingerprint for each host in the network.

Up to 15 attributes can be obtained from the captured data using Suricata and port mirroring [10]. However, when DNS logs are used as input, IP-based attributes such as the number of cities of resolved IP addresses and the number of countries of resolved IP addresses cannot be captured. As a result, 13 attributes are obtained that allow denoting behavior of bot-infected hosts. The attributes are generated by the flow of DNS events which are used to create the footprints. The fingerprints are generated for each active host, and intervals of one hour for ten days are considered. Each of the attributes is presented below with its respective description.

3.3.1. Attributes based on DNS requests

- **Number of DNS requests per hour** (P_1): It gives an early insight into whether or not a host is infected. Bot-infected machines tend to have more requests per hour than usual.
- **Number of distinct DNS requests per hour** (P_2): DGA malware-infected hosts tend to have more distinct requests than regular hosts.
- **Increased number of requests for a single domain** (P_3): It helps to detect the existence of a DNS tunnel in which confidential information is transferred through this protocol.
- **Average number of requests per minute** (P_4): It helps to detect malware-infected machines that do not use short bursts of DNS requests but regularly contribute to DNS requests through a suspend interval. It is calculated by dividing the number of requests sent by the host's time and using the domain name resolution service.
- **Highest number of requests per minute** (P_5): It helps to detect malware-infected bots that use a short burst of DNS requests to communicate with the C&C server. The DGA algorithms generate multiple URLs.
- **Number of MX record queries** (P_6): It is an indicator of spam-based botnets on the network.
- **Number of PTR record queries** (P_7): It helps to detect hosts with abnormal behavior on a network and possible infection.
- **Number of different DNS servers consulted** (P_8): It helps to detect machines with abnormal behavior on the network. It is rare for a standard system to query more than one DNS server.

3.3.2. Domain-based attributes

- **Number of different TLDs queried** (P_9): It effectively detects DGA-based bots that generate not only random domains with different SLDs but also different TLDs.
- **Number of different SLDs consulted** (P_{10}): It is an indicator of the presence of DGA-based bots on the network.

Table 2
Example of DNS fingerprint data set obtained from classifier module.

@timestamp	IP	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P15
2022-07-12T20:00	190.x.x.x1	235737	11762	1877	7604.41	9555	0	3184	1	201	2652	20.04	442	0.058
2022-07-12T20:00	190.x.x.x2	204655	6306	12440	6601.77	9877	1	364	1	168	1964	32.45	76	0.034
2022-07-12T20:00	190.x.x.x3	120977	6062	11306	3902.48	5309	0	251	1	156	1999	19.95	100	0.054
2022-07-12T20:00	190.x.x.x4	116319	6501	8619	3752.22	5377	0	415	1	137	1536	17.89	59	0.058

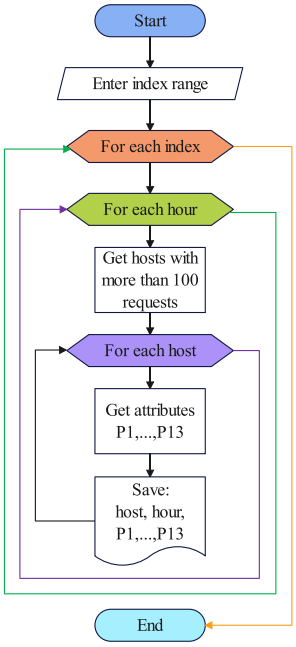


Fig. 4. Flowchart for obtaining the attributes to generate the DNS fingerprints.

- **Uniqueness ratio** (P_{11}): It is the ratio between the number of requests sent and the number of different requests sent, assuming that the host has sent at least 100 requests per hour.

3.3.3. Attributes based on response

- **Number of failed queries/NXDOMAIN** (P_{12}): It is an indicator of host infection on the network. In domains generated by DGA, of several queries made, only one or none is resolved. In this way, many failed queries are generated, which generates a response code equal to NXDOMAIN.

3.3.4. Attributes based on mapping (FQDN-IP)

- **Flow ratio** (P_{13}): It is the proportion of the different requests sent to the different resolved IP addresses, provided that the host has sent at least 100 queries and has received at least 100 responses.

The Python programming language is used to access Elasticsearch as an external client. Elasticseach's search and analytics engine obtains the DNS attributes required for each host; this will generate the DNS fingerprints of the hosts for hours. Fig. 4 presents the flowchart to generate the fingerprints; it performs queries from Python to Elasticsearch.

An example of a data set obtained from the DNS fingerprint generation component is illustrated in Table 2. Each line represents a host's fingerprint and has the date, the IP, and the attributes (P_1 to P_{13}); the date and the IP have been added for better visualization and analysis. Thus, each line is a unique host fingerprint for a specific date and time.

3.4. Anomaly detection

With a DNS fingerprint data set of hosts, the search for an anomaly detection engine classifies the fingerprints. The classification will be in two categories: bot or clean, depending on the presence or absence of an anomaly. The amount of benign traffic that is greater than the amount of malicious traffic must be considered a premise for network anomaly detection. The classification thresholds will vary depending on the nature of the network on which the system is implemented. So, the anomaly detection engine will do it with unsupervised machine learning.

The scikit-learn machine learning library in Python presents a section dedicated to outlier and novelty detection. In the context of outlier detection, anomalies/outliers cannot form a dense group. Therefore, the estimators assume that outliers/anomalies are in low-density regions [37]. The Isolation Forest algorithm has been used, which scikit-learn developers recommend due to its efficiency for moderately high-dimensional data sets; in this case, 13.

3.4.1. Domain name analysis: Randomness detection in domain names

Most DGA-based domain names contain meaningless strings that are difficult to pronounce or read due to their generation's algorithmic nature and the avoidance of matching existing domain names. The existence of consonants or sequential vowels (considering the letter "y" as a vowel) and a high entropy are presented as characteristics of randomness in a domain [10,12,13].

The Shannon's entropy H can be calculated as $H = -\sum_{i=1}^{n} p_i \log p_i$, where p_i is an element of the probability distribution $P_n = (p_1, \ldots, p_n)$ with $p_i \geq 0$ for $i = 1, \ldots, n$ and $\sum_{i=1}^{n} p_i = 1$.

Algorithm 1 measures the randomness of domain names. This algorithm will be used as a support tool applied to the fingerprints detected as abnormal. It seeks to determine queries to domains that follow randomness patterns typical of domains generated by DGA. A filter is made of those domains with no response (NXDOMAIN) to reduce the number of verified domains. The hypothesis is that a bot generates a flood of queries with DGA-based domains until one of these is resolved. We use the Algorithm 1 and the thresholds established in [10] for entropy values H, the maximum number of sequential vowels SV, and the maximum number of sequential consonants SC. High accuracy for classifying domain names as usual or DGA-based is achieved. One of the future works is to improve this algorithm.

Algorithm 1: Randomness Measuring Algorithm

Input : Domain name
Output : *Normal Domain Name* **or** *DGA-based Domain Name*
Parameters: H, domain's length, SC, SV

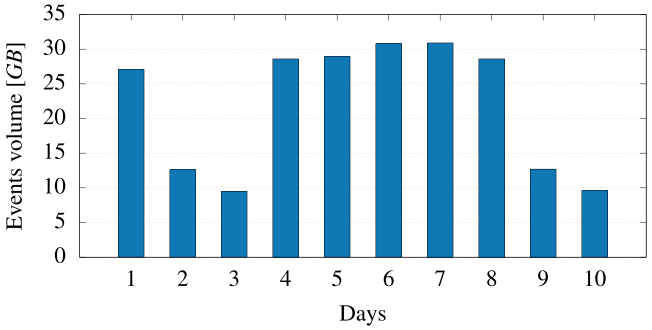
```
if (H ≤ 2) and (domain's length < 5) then
    return Normal domain name;
else
    if (H ≤ 3.24) then
        return Normal domain name;
    else
        if (SC < 4) or (SV < 4) then
            return Normal domain name;
        else
            return DGA-based domain name;
        end
    end
end
```

3.5. Classifier for autonomous detection

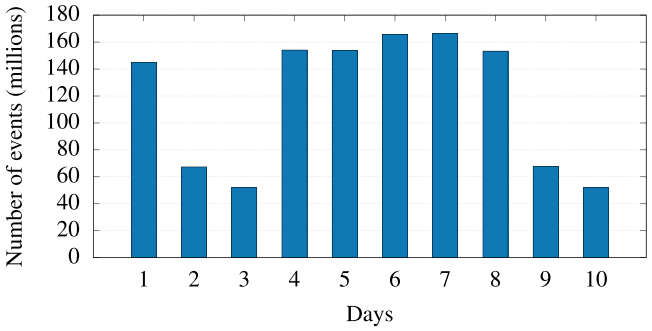
The output of the anomaly module presents a data set with data labeled as bot or clean. This data set enables the ability to train a

Table 3
Minimum, average, and maximum values of the DNS attributes (Att) of the fingerprints.

Att	Description	Minimum	Average	Maximum
P_1	Number of DNS requests per hour	100	456.371	463930
P_2	Number of different DNS requests per hour	1	57.689	53908
P_3	Most requests for a single domain per hour	2	75.923	183248
P_4	Average number of requests per minute	1.695	13.418	7605.41
P_5	Most requests per minute	2	54.12	15111
P_6	Number of MX record queries per hour	0	0	2869
P_7	Number of PTR record queries per hour	0	8.615	87555
P_8	Number of different DNS servers queried per hour	1	1	4
P_9	Number of different TLD domains queried per hour	0	6	329
P_{10}	Number of different SLD domains consulted per hour	0	27.786	7063
P_{11}	Uniqueness ratio per hour	1.023	5.945	18708
P_{12}	Number of failed/NXDOMAIN queries per hour	0	0	10160
P_{13}	Hourly flow rate	0.0	0.246	71.0



(a) Volume of events generated



(b) Number of DNS events generated

Fig. 5. Captured DNS events.

supervised machine learning algorithm. The purpose is to have an autonomous system for detecting future infected hosts on the network in less time. The Random Forest algorithm [3] was used to train and test the data in the classifier module.

4. Results and analysis

4.1. Captured data

The data is captured following the procedure presented in Section 3.3 for ten days. Fig. 5(a) depicts the volume (obtained from Kibana) of the DNS indexes generated by days, which add up to 52.4 GB. Fig. 5(b) illustrates the number of DNS events generated from a quantity perspective. There are peaks of around 8 million events in 3 h, yielding a total of 687,526,874 events during the ten days of data capture. These events will allow generating of the hosts' DNS fingerprints per hour.

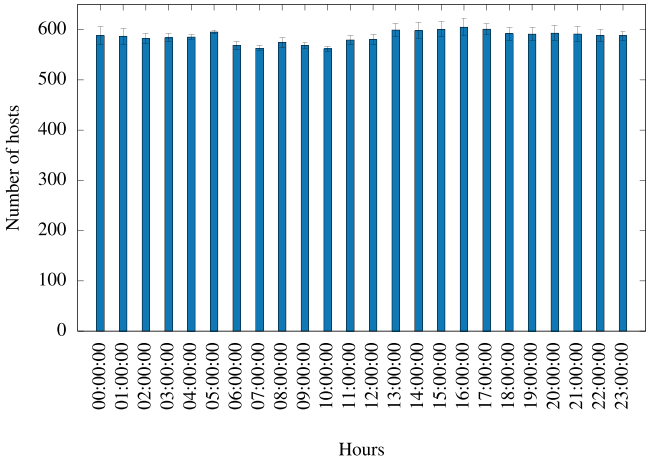


Fig. 6. Number of active hosts on the network per hour.

4.2. Generation of DNS fingerprints

The application of the DNS fingerprint generator module allows obtaining values of interest, such as the number of active hosts on the network; its behavior per hour is shown in Fig. 6. The minimum, maximum, and average data captured by the hosts on the network are 578, 1149, and 705, respectively. Hosts that meet the cache condition and have at least 100 queries per hour will generate the fingerprint data set.

The obtained fingerprints are added to a new index in Elasticsearch. The tool counts 86,993 fingerprints generated for the ten days of traffic capture. The traffic was generated by 4687 hosts interacting on the network and meeting the project conditions. Table 3 shows each attribute's minimum, average, and maximum values.

4.3. Anomaly detection

The number of trees that best fits the data can be defined from the parameters of the Isolation Forest algorithm. The relationship between tree growth and computational complexity is directly proportional. Fig. 7 depicts the amount of data classified as anomalies as a function of the number of trees from 2 to 1000. A trend of difficulty in isolating anomalous values is observed as the number of trees in the model increases. The number of anomalies found ranges from 5694 to 10,089.

The experiment considers 70 as the number of trees in the model. It is the first peak when the number of anomalies is 6303. From a cybersecurity point of view, it is better to have some false positive detection than to ignore possible intrusions. Another parameter of the

Table 4
Some hosts and their percentage of footprints cataloged as bots.

IP	# Bot footprints	# Total footprints	Percentage
190.15.X.X	52	52	100
190.15.X.X	240	240	100
190.15.X.X	240	240	100
45.182.X.X	240	240	100
192.188.X.X	240	240	100
201.159.X.X	240	240	100
192.188.X.X	229	240	95
143.255.X.X	205	240	85
201.159.X.X	192	240	80
143.255.X.X	166	240	69
201.159.X.X	148	240	61
190.96.X.X	128	240	53
192.188.X.X	123	240	51
200.12.X.X	117	240	48
201.234.X.X	92	193	47
190.15.X.X	104	240	43
192.188.X.X	104	240	43
45.235.X.X	101	240	42
190.15.X.X	98	240	40
192.188.X.X	88	234	37
192.100.X.X	84	225	37

Table 5
Footprints.

Type	# Footprints	Percentage
Clean	81462	93.6%
Bot without DGA	5531	6.4%
Bot with DGA	323	0.003%

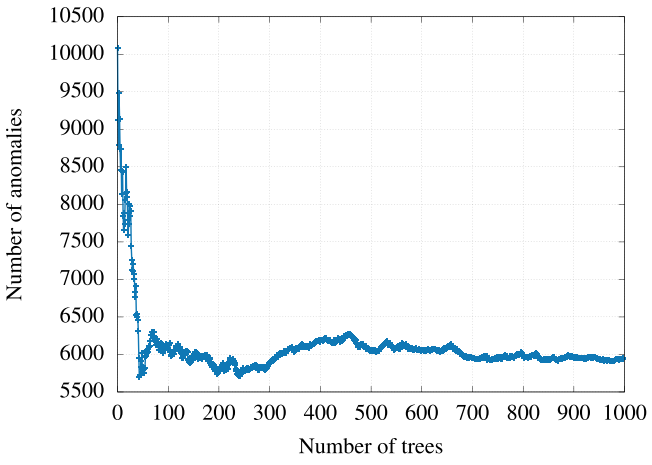


Fig. 7. Number of samples classified as anomalies based on the number of trees.

algorithm is the contamination, which is set to automatic so that the algorithm draws its threshold between normal and abnormal [38].

The footprints cataloged as a bot and clean based on their attributes has been visualized. Fig. 10 shows the classification of the footprints according to the attributes (from 10a to 10m) considering a logarithmic scale. This scale allows us to observe how the highest density of points are clean footprints and based on this dense area; the algorithm draws thresholds. Values that exceed the thresholds will be classified as bot. An analysis of the numerical values of the footprint attributes was previously carried out in Table 3. The graphs that stand out are Fig. 10f and Fig. 10h, corresponding to the number of queries to MX records and the number of different DNS servers consulted. The P_6 attribute's average value is 0, so only the anomalous values are highlighted in red. For attribute P_8, its value for all traces is 1; all hosts are making their queries to the local DNS server. For this reason, the unique value is presented as a line.

Other values of interest are the classification percentages based on the total footprints. Table 5 presents the values for the number of footprints classified as clean and bot, the percentage for the total number of footprints, and the number of hosts within this classification.

The system can compare the total footprints a host has generated against the footprints of the same host classified as anomalies. A fingerprint is considered to represent the behavior of a host at a specific time. In this context, a host with many footprints classified as bad denotes that its general behavior over time only differs from most hosts on the network. Thus, their behavior will always be abnormal. Table 4 shows some hosts with the most traces cataloged as anomalous and their percentage compared to the total number of traces generated. It is observed as is the case of the hosts 190.15.X.X and 45.182.X.X whose percentage of footprints cataloged as anomalous is 100%. Some IPs can be servers; internal networks are behind these addresses, which would explain their discordant number of requests compared to an IP associated with a host. This analysis allows us to consider taking some action on these IPs to have more homogeneous data.

We can perform a visual comparison of a clean host and an infected host at the same time. From the general point of view of net traffic, an anomaly may not be visible, as shown in Figs. 11(a) and 11(b). However, the infected hosts' detection method allows for finding anomalies in the same data set by analyzing them individually. Figs. 11(c) and 11(d) illustrate the DNS traffic of a host classified as clean, which denotes sequentiality in its queries. In Figs. 11(e) and 11(f) the DNS traffic of a host cataloged as a bot is shown, and its query pattern denotes some communication peaks in bursts. Cases A and B presented in Fig. 11 present a comparison of the traffics: net, of a clean host and of the hosts 190.15.X.X and 190.15.X.X cataloged as bot, for the times 02/17/2021: 08: 00: 00 and 02/11/2021: 16: 00: 00, respectively.

4.4. Verification of queries to DGA domains

The failed queries are searched for those whose SLDs have randomly generated characteristics for the traces cataloged as anomalous. In [10], authors evaluated the randomness meter with a data set of DGA family domains and around the first 200 thousand domains of the Alexa query ranking. However, in an internal network, there may be own domain names that the algorithm has classified as suspicious and, at first glance, appear legitimate. The human factor is necessary to analyze

Table 6
Suspicious SLD domain names for some IPs.

IP	SLD
192.188.X.X	cgablcehfnrdpzi, epmsqkjbkgwnv, w42f4ctqv4, chxvlothvb, othvbyopsakbg, uyrrfqqbodu, lnyrczvmoyzro, crwdcntrl, xnycmjmulrsopfmm3x, brtqvyhbpskluz, zqnvkbwhhfwpf, dbankcdn
200.7.X.X	tdfilrvxsqwzli, zdwlgulhfflczk, uyqeztwyfiyusx, wkpzxttbaw, rmrjalyqztfbn, ywarpjtnr, ztsvvxmuohahskz, uyteuylflbiey, wlhhspuscfrg, loswletnvlmv, nmethhafuqgl, qdzpkxrdnkn, tynswlnduizdjw, geqjcjjvx, ujjqlcvehuj
190.15.X.X	zhuvcruhoolkegc, elkbphh, nkgfcdvzyuq, ttvnw, uihcazcnvwey, hdltormauqgmff, ryrunxfjfah, zthhymxm, wmvfuemoczf, gnebxwcinkcma, szawvtxekdovej, xubytfhk, hzujwglxdbt

Table 7
Confusion matrix for the Random Forest model with 25 trees.

Real/Predicted	Bot	Clean
Bot	367	33
Clean	15	8867

Table 8
Classification report for the Random Forest model with 25 trees.

Parameters	Precision	Sensivity	F1-score	Support
Bot	0.98	0.96	0.97	1726
Clean	1.00	1.00	1.00	22016
accuracy			1.00	23742
macro avg	0.99	0.98	0.98	23742
weighted avg	1.00	1.00	1.00	23742

the existence of domains with a random character; this would be a new filter that supports classifying a host as infected. Thus, Table 6 shows the SLDs' extraction for some fingerprints that were classified as a bot and had more than ten not resolved domains with DGA characteristics. This behavior is repetitive in time for the footprints associated with the hosts in the table, with new groups of random SLDs appearing each time (see Fig. 8).

4.5. Machine learning

The number of trees in the Isolation Forest algorithm determines the learning precision of the algorithm. Fig. 9 illustrates the impact of the number of trees with two metrics: the classification precision and the processing time required to classify 9282 footprints that represent the 30% of the data set.

It can be observed how increasing the number of trees in the model improves the accuracy of the classification algorithm; however, this increment is not linear due to the random nature of the algorithm. Fig. 9 also presents the processing time of the algorithm as a function of the number of trees. Again, a direct relationship is observed between the number of trees in the model and the training time; the fewest trees used in the model have the lowest computational cost for a similar accuracy.

The precision and processing time values are considered a function of the number of trees. For this experiment, we consider the accuracy of 0.995 in the model, achieved using 25 trees. Table 7 details the confusion matrix obtained. The false-positive rate is low, considering a large number of fingerprints correctly classified as clean. The false-negative rate is higher compared to the number of bot footprints. An active host on the network generates a large number of footprints. Therefore, if there is a bot infection with the algorithm's accuracy rate, the probability that at least one fingerprint (associated with a *host-bot*) will be cataloged correctly is high. With this analysis, the performance of this module is considered satisfactory.

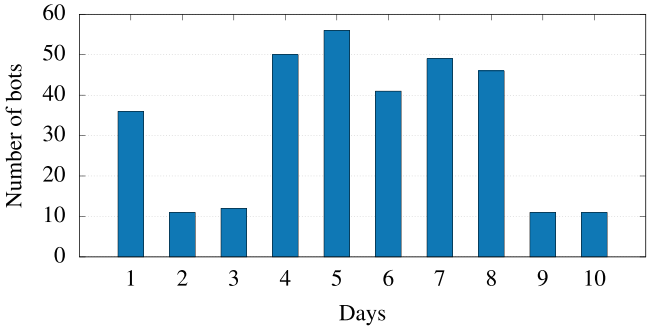


Fig. 8. Detected bots by day.

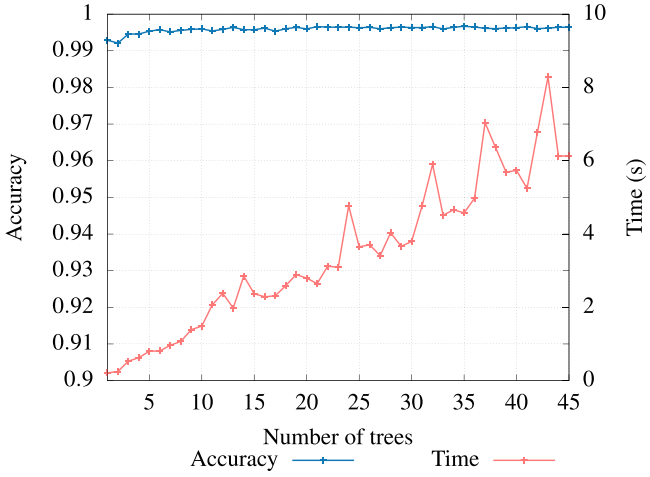
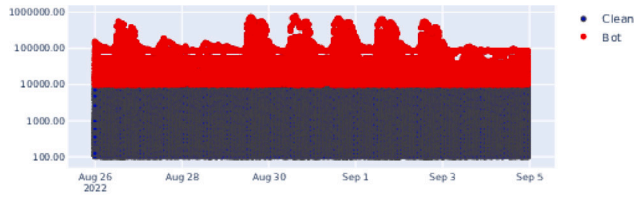


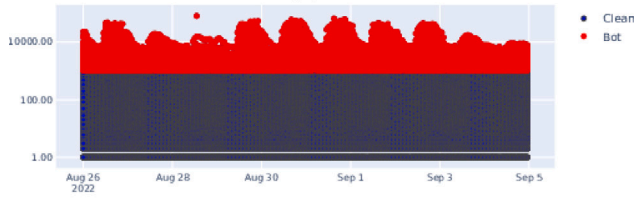
Fig. 9. Precision parameters and training time as a function of the number of trees for the Random Forest algorithm.

It is worth mentioning that the results obtained may vary depending on the nature of the network on which it is working. Thus, what could turn out to be an irrelevant attribute for the traffic of this project could be an attribute of great importance for detecting bot infections in another network.

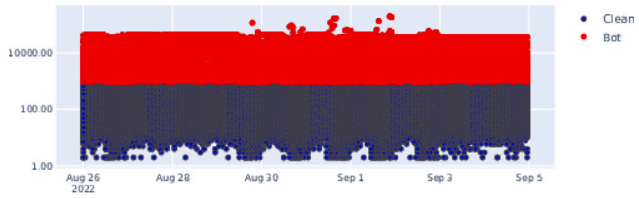
With the values of the confusion matrix, statistical weights have been obtained that help understands the algorithm's quality. Table 8 reports precision, sensitivity, and F1 score values. The system trained the detection model with many footprints (the number of clean footprints is much more significant than the bot footprints); the algorithm performs better for fingerprints whose behavior is within normal limits.



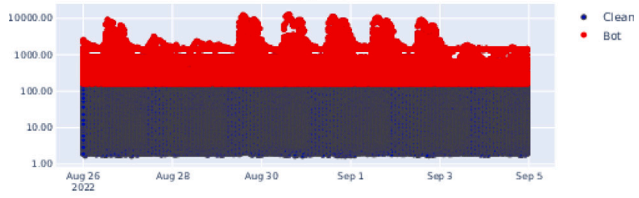
(a) P_1



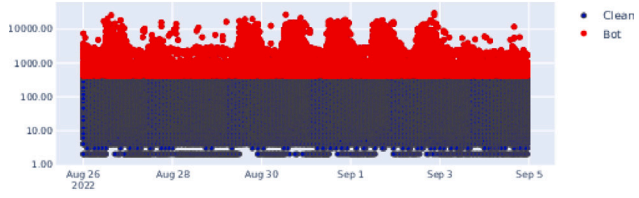
(b) P_2



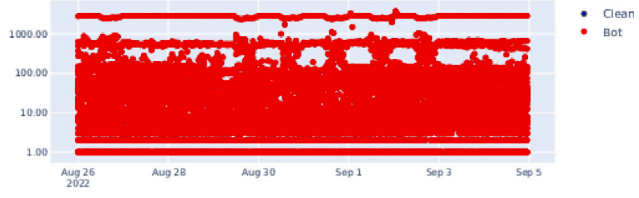
(c) P_3



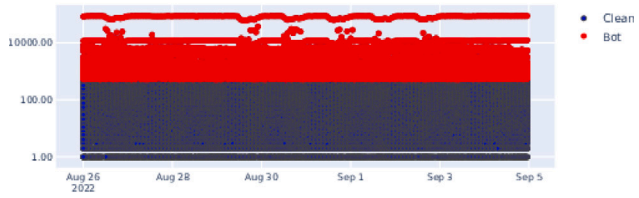
(d) P_4



(e) P_5



(f) P_6

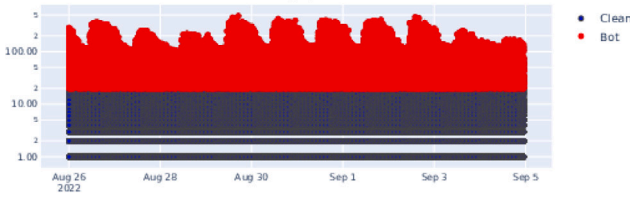


(g) P_7

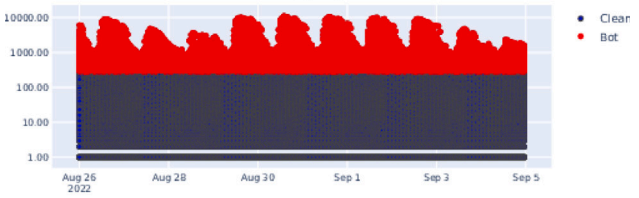
Fig. 10. Classification of footprints based on their attributes on a logarithmic scale.

Scikit Learn allows us to break down the importance of each parameter for learning the model developed. Table 9 shows the importance values (Weight) of the parameters for the algorithm. The most important parameter to classify a footprint as a bot or clean is the number

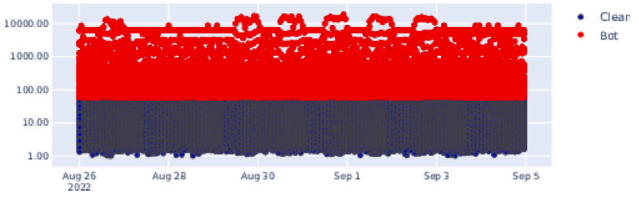
(h) P_8



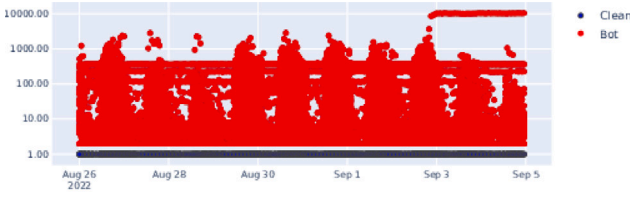
(i) P_9



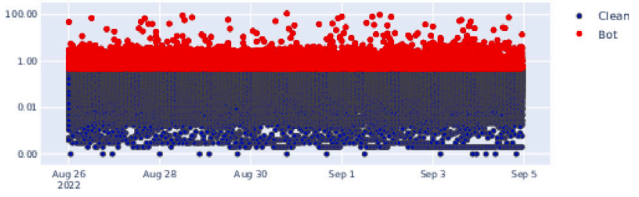
(j) P_{10}



(k) P_{11}



(l) P_{12}



(m) P_{13}

Fig. 10. (*continued*).

of different SLD domains a host consults per hour. Next, there are parameters such as the number of different requests per hour, total requests per hour, and average requests per minute; these parameters are related to the number of requests a host makes. The parameters referring to the geographic locations of the resolved domains are also relevant to the model. A similar percentage of importance has the highest number of requests for a single domain, the number of PTR queries per hour, the highest number of requests per minute, and the number of failed queries. The irrelevant attribute is the number of different servers consulted, whose value was the unit for all the tracks. For this architecture, no-host tried communicating with another DNS server other than the local one.

(a) Net traffic case A (b) Net traffic case B

(c) Clean traffic case A (d) Clean traffic case B

(e) Bot traffic case A (f) Bot traffic case B.

Fig. 11. DNS traffic cases A and B for net, clean, and bot.

Table 9
Importance of the attributes (Att) for the Random Forest model with 25 trees.

Att	Description	Weight
P_4	Average number of requests per minute	0.137
P_9	Number of different TLD domains queried per hour	0.113
P_{10}	Number of different SLD domains consulted per hour	0.11
P_7	Number of PTR records queries per hour	0.096
P_2	Number of different DNS requests per hour	0.083
P_3	Highest number of requests for a single domain per hour	0.064
P_{11}	Uniqueness ratio per hour	0.061
P_{12}	Number of failed / NXDOMAIN queries per hour	0.046
P_6	Number of MX record queries per hour	0.04
P_5	Most requests per minute	0.035
P_8	Number of different DNS servers queried per hour	0.022
P_{13}	Hourly flow rate	0.019

5. Conclusions

Detecting DNS traffic anomalies in the network allows for defining a methodology to detect bot-infected hosts. A botnet's life cycle depends on the DNS service as a tool to resolve domains of C&C servers and establish connections with bots. It also allows to migration of the IP addresses of the C&C servers and avoids blacklists. Current research fields for DNS-based botnet detection focus on the bot-infected host and DGA-based detection techniques. This project addressed these techniques, the first by creating fingerprints of the hosts based on 13 DNS parameters, and the second, through an algorithm for measuring the randomness of a domain based on the entropy value and the number of vowels and sequential consonants.

The proposed architecture allows a system implementation that captures and generates events using a Graylog and Logstash server. The system processes the captured events offline. Thus, the system does not represent any threat to the regular operation of the network. The results obtained do not contemplate a corrective action on the network. Instead, the project is presented as an alert system for detecting anomalies in DNS events that denote activities related to bot infection in hosts.

The integration of the tools and processes allowed complete management of DNS events of a production network. Graylog and Logstash proved to be powerful tools for capturing traffic and generating DNS events, reducing the need for direct interaction with the raw flow of the network. Integrating the ELK stack as SIEM allowed better management of DNS events, such as eliminating irrelevant fields and dividing domains into levels. Integrated fields proved to be essential attributes when training a machine learning model. Using Python, it was possible to take advantage of the Elasticsearch search and analytics engine and generate DNS fingerprints based on stored events.

The conducted methodology contemplates extracting 13 DNS parameters per hour from the hosts to generate fingerprints and applying ML to detect anomalies. In 10 days 241 hosts generated 687,526,874 events that led to the creation of 32,182 DNS fingerprints. The Isolation Forest model with 110 trees classified 2287 as anomalous, representing 7% of total footprints belonging to 58 hosts. A Random Forest model was trained using 25 trees with the classified tracks, achieving an accuracy of 0.995.

It was observed as the method of detecting infected bot hosts. It detects traffic anomalies that do not appear when analyzing the total network traffic and are a typical pattern of bot communications. Also, applying a domain randomness detection algorithm on the footprints classified as bots has been allowed to verify the existence of several failed queries, such as bursts. Hosts made queries to domain names with random characteristics, typical of DGA bots.

The model generated by Random Forest provided information on the level of importance each feature of the footprints has for detecting footprints belonging to bots. This information is consistent with the theory. A bot's communication search with its botmaster generates increases in DNS queries on an infected host. In turn, in an attempt to evade security elements, variety is generated in domain names. The machine learning algorithm is sensitive to these parameters, and due to its high precision, the system is considered efficient for detecting future bot-infected hosts in said network.

6. Future work

CEDIA corporation manages the computational and network resources to expand the system's application field within a framework of tool development focused on cybersecurity. Thus, this work opens the way to projects around log analysis as a security intrusion detection method. At the same time, it allows working on a real-world network in a stable platform for capturing and processing network events.

Our solution allows for implementing more DNS event generation and traffic analysis points to enrich the fingerprint data set. The system's distributed architecture permits having several points sending

DNS logs as event generators. These events can be sent to Logstash through agents like Filebeat. This solution is applicable to those hosts whose behavior is always abnormal. Thus, this host could be replaced by the set of hosts it represents.

Also, we can generate a whitelist of internal domains. In this way, applying the domain randomness detection algorithm reduces its false-positive rate; this improves the detection technique for bots based on DGA to help enhance the infection detection rate. We plan to explore other security intrusion detection techniques based on log analysis. The same framework can be used to detect crypto miners.

On the other hand, the proposed framework is implemented only using machine learning. However, the idea is to implement different techniques, including deep learning. We want to compare the current implementation with, for example, generative adversarial networks (GAN) [39,40].

CRediT authorship contribution statement

Vicente Quezada: Conceptualization, Methodology, Software, Writing – original draft, Data curation. **Fabian Astudillo-Salinas:** Supervision, Writing – review & editing, Software, Investigation, Visualization. **Luis Tello-Oquendo:** Investigation, Writing – review & editing. **Paul Bernal:** Data curation, Validation.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Fabian Astudillo-Salinas reports equipment, or supplies was provided by CEDIA.

Data availability

The authors do not have permission to share data

References

[1] T.S. Wang, H.T. Lin, W.T. Cheng, C.Y. Chen, DBod: Clustering and detecting DGA-based botnets using DNS traffic analysis, Comput. Secur. 64 (2017) 1–15, http://dx.doi.org/10.1016/j.cose.2016.10.001.

[2] M. Singh, M. Singh, S. Kaur, Issues and challenges in DNS based botnet detection: A survey, Comput. Secur. 86 (2019) 28–52, http://dx.doi.org/10.1016/j.cose.2019.05.019.

[3] M. Singh, M. Singh, S. Kaur, Detecting bot-infected machines using DNS fingerprinting, Digit. Investig. 28 (2019) 14–33, http://dx.doi.org/10.1016/j.diin.2018.12.005.

[4] M. Singh, M. Singh, S. Kaur, Identifying bot infection using neural networks on DNS traffic, J. Comput. Virol. Hacking Tech. (2023) 1–15.

[5] M. Nazari, Z. Dahmardeh, S. Aliabady, A novel approach of botnets detection based on analyzing dynamical network traffic behavior, SN Comput. Sci. 2 (4) (2021) 1–11.

[6] K. Alieyan, A. Almomani, M. Anbar, M. Alauthman, R. Abdullah, B.B. Gupta, DNS rule-based schema to botnet detection, Enterprise Inf. Syst. 15 (4) (2021) 545–564.

[7] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, D. Garant, Botnet detection based on traffic behavior analysis and flow intervals, Comput. Secur. 39 (2013) 2–16.

[8] C.-M. Chen, H.-C. Lin, Detecting botnet by anomalous traffic, J. Inf. Secur. Appl. 21 (2015) 42–51.

[9] R. Sharifnya, M. Abadi, Dfbotkiller: Domain-flux botnet detection based on the history of group activities and failures in dns traffic, Digit. Investig. 12 (2015) 15–26.

[10] A.O. Almashhadani, M. Kaiiali, D. Carlin, S. Sezer, MaldomDetector: A system for detecting algorithmically generated domain names with machine learning, Comput. Secur. 93 (2020) http://dx.doi.org/10.1016/j.cose.2020.101787.

[11] K. Alieyan, A. Almomani, A. Manasrah, M.M. Kadhum, A survey of botnet detection based on DNS, Neural Comput. Appl. 28 (7) (2017) 1541–1558, http://dx.doi.org/10.1007/s00521-015-2128-0.

[12] D.K. Vishwakarma, Domain Name Generation Algorithms, Masaryk University, 2017.

[13] A.O. Almashhadani, M. Kaiiali, S. Sezer, P. O'Kane, A multi-classifier network-based crypto ransomware detection system: A case study of Locky Ransomware, IEEE Access 7 (2019) 47053–47067, http://dx.doi.org/10.1109/ACCESS.2019.2907485.

[14] Y. Xing, H. Shu, H. Zhao, D. Li, L. Guo, Survey on botnet detection techniques: Classification, methods, and evaluation, Math. Probl. Eng. 2021 (2021) 1–24.

[15] J. Yadav, J. Thakur, Botnet: Evolution life cycle architecture and detection techniques, Mukt Shabd J. 9 (6) (2020) 4265–4281.

[16] A. Muhammad, M. Asad, A.R. Javed, Robust early stage botnet detection using machine learning, in: 2020 International Conference on Cyber Warfare and Security, ICCWS, IEEE, 2020, pp. 1–6.

[17] R.S.S. Moorthy, N. Nathiya, Botnet detection using artificial intelligence, Procedia Comput. Sci. 218 (2023) 1405–1413.

[18] N.V.T. Hiep, T.V. Nikolaevich, D.M. Tuan, N.T. Lam, N.A. Tuan, Detecting botnet based on network traffic, Int. J. Adv. Trends Comput. Sci. Eng. 9 (3) (2020) 3010–3014, http://dx.doi.org/10.30534/ijatcse/2020/79932020.

[19] A. Satoh, Y. Nakamura, D. Nobayashi, T. Ikenaga, Estimating the randomness of domain names for DGA bot callbacks, IEEE Commun. Lett. 22 (7) (2018) 1378–1381, http://dx.doi.org/10.1109/LCOMM.2018.2828800.

[20] A.D. Biradar, B. Padmavathi, BotHook: A supervised machine learning approach for botnet detection using DNS query data, in: ICCCE 2019: Proceedings of the 2nd International Conference on Communications and Cyber Physical Engineering, Springer, 2020, pp. 261–269.

[21] J. Wu, Artificial neural network based DGA botnet detection, J. Phys.: Conf. Ser. 1578 (1) (2020) 012074.

[22] A. Soleymani, F. Arabgol, A novel approach for detecting DGA-based botnets in DNS queries using machine learning techniques, J. Comput. Networks Commun. 2021 (2021) 1–13.

[23] J. Seo, S. Lee, Abnormal behavior detection to identify infected systems using the apchain algorithm and behavioral profiling, Secur. Commun. Netw. 2018 (2018) http://dx.doi.org/10.1155/2018/9706706.

[24] D. Wook Kim, J. Zhang, Deriving and measuring DNS-based fingerprints, J. Inf. Secur. Appl. 36 (2017) 32–42, http://dx.doi.org/10.1016/j.jisa.2017.07.006.

[25] M. Singh, Anomaly based Botnet Detection using DNS Traffic Analysis (Ph.D. thesis), Thapar Institute of Engineering & Technology, URL http://hdl.handle.net/10266/5959.

[26] M. Asadi, M.A.J. Jamali, S. Parsa, V. Majidnezhad, Detecting botnet by using particle swarm optimization algorithm based on voting system, Future Gener. Comput. Syst. 107 (2020) 95–111.

[27] A. Moubayed, M. Injadat, A. Shami, Optimized random forest model for botnet detection based on DNS queries, in: 2020 32nd International Conference on Microelectronics, ICM, IEEE, 2020, pp. 1–4.

[28] A.M. Manasrah, W.B. Domi, N.N. Suppiah, Botnet detection based on DNS traffic similarity, Int. J. Adv. Intell. Paradigms 15 (4) (2020) 357–387.

[29] S. Al-Mashhadi, M. Anbar, I. Hasbullah, T.A. Alamiedy, Hybrid rule-based botnet detection approach using machine learning for analysing DNS traffic, PeerJ Comput. Sci. 7 (2021) e640.

[30] C.-I. Fan, C.-H. Shie, C.-M. Hsu, T. Ban, T. Morikawa, T. Takahashi, IoT botnet detection based on the behaviors of DNS queries, in: 2022 IEEE Conference on Dependable and Secure Computing, DSC, IEEE, 2022, pp. 1–7.

[31] Graylog, What is Graylog? 2022, URL https://go2docs.graylog.org/5-0/what_is_graylog/what_is_graylog.htm.

[32] elastic, Elastic documentation, 2022, URL https://www.elastic.co/.

[33] Python, Python tutorial, 2022, URL https://docs.python.org/3/tutorial/index.html.

[34] StamusNetworks", StamusNetworks/SELKS, 2018, URL https://github.com/StamusNetworks/SELKS.

[35] J. García Merino, Ventajas e implementación de un sistema SIEM, 2018, URL http://hdl.handle.net/10609/95287.

[36] S. Gómez Fernández, Implementación de un IDS de bajo coste para uso doméstico o en la pequeña empresa, 2019.

[37] Scikit-learn, 2.7. Novelty and outlier detection, 2020, URL https://scikit-learn.org/stable/modules/outlier_detection.html.

[38] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation-based anomaly detection, ACM Trans. Knowl. Discov. Data (TKDD) 6 (1) (2012) 1–39.

[39] C. Yin, Research on Network Anomaly Detection Technology Based on Deep Learning, University of Information Engineering, Strategic Support Forces, Zhengzhou, China, 2018.

[40] X. Xia, X. Pan, N. Li, X. He, L. Ma, X. Zhang, N. Ding, GAN-based anomaly detection: A review, Neurocomputing (2022).

Vicente Quezada received his degree in Electronics and Telecommunications Engineering from the University of Cuenca, Ecuador, in 2021. His current research interests include Cybersecurity, Information Security and Data Analysis.

Fabian Astudillo-Salinas received the B.S.E (C.S) degree from Universidad de Cuenca, Cuenca, Ecuador, in 2007, and the M. S. and Ph.D. degrees from the Institut National Polytechnique de Toulouse, Toulouse, France, in 2009 and 2013, respectively. Since 2013, he has been a Full-Time Researcher with the Department of Electrical, Electronic, and Telecommunications Engineering, Universidad de Cuenca, Cuenca, Ecuador. His research interests include network coding, wireless sensor networks, vehicular networks, networked control systems, simulation of networks, performance of networks, cybersecurity and HPC.

Luis Tello-Oquendo received the electronic and computer engineering degree (Hons.) from Escuela Superior Politécnica de Chimborazo (ESPOCH), Ecuador, in 2010, the M.Sc. degree in telecommunication technologies, systems, and networks, and the Ph.D. degree (Cum Laude) in telecommunications from Universitat Politécnica de Valéncia (UPV), Spain, in 2013 and 2018, respectively. From 2013 to 2018 he was Graduate Research Assistant with the Broadband Internetworking Research Group, UPV. From 2016 to 2017 he was a Research Scholar with the Broadband Wireless Networking Laboratory, Georgia Institute of Technology, Atlanta, GA, USA. He is currently an Associate Professor with the Universidad Nacional de Chimborazo. His research interest include MTC, wireless SDN, 5G and beyond cellular systems, IoT, machine learning. He is a member of the IEEE and ACM. He received the Best Academic Record Award from the Escuela Técnica Superior de Ingenieros de Telecomunicación, UPV, in 2013, the Extraordinary Doctoral Thesis Award from the Escuela de Doctorado, UPV, in 2019, and the Best Researcher Award from the Ecuadorian Corporation for the Development of Research and the Academy (CEDIA), in 2021.

Paul Bernal is a former CSIRT Engineer at the Ecuadorian NREN. Specialized in processing & managing IT Security Incidents & Data since 2013. Worked in troubleshooting, optimizing & hardening Linux Servers, networks and Open Source services since early 2000s. In 2002 co-founded and ran a hosting company for ~10 years, providing shared, virtual and dedicated servers/services with LAMP mostly. Pretty much since graduating, been an instructor in Open-Source IT & Cybersecurity areas at continuing education and Masters degree programs. Linux and OpenSource advocate since 1998 and a Programmer since the 80s, starting with Atari BASIC, now mainly PHP/Bash.