Detecting Anomalies at a TLD Name Server Based on DNS Traffic Predictions

Diego Madariaga, Javier Madariaga, Martín Panza, Javier Bustos-Jiménez, and Benjamin Bustos

***Abstract*—The Domain Name System (DNS) is a critical component of Internet infrastructure, as almost every activity on the Internet starts with a DNS query. Given its importance, there is increasing concern over its vulnerability to attacks and failures, as they can negatively affect all Internet-based resources. Thus, detecting these events is crucial to preserve the correct functioning of all DNS components, such as high-volume name servers for top-level domains (TLD). This paper presents a near real-time Anomaly Detection Based on Prediction (AD-BoP) method, providing a useful and easily explainable methodology to effectively detect DNS anomalies. AD-BoP is based on the prediction of expected DNS traffic statistics, and could be especially helpful for TLD registry operators to preserve their services' reliability. After an exhaustive analysis, AD-BoP is shown to improve the current state-of-the-art for anomaly detection in authoritative TLD name servers.**

***Index Terms*—Domain Name System, anomaly detection, DNS traffic, top-level domain, prediction, unsupervised learning.**

I. Introduction

WITH the continuous growth of the Internet and its increasing number of users during the last decades, the Domain Name System (DNS) has also experienced an important evolution. DNS has become a vital part of the Internet itself, as it is responsible for the translation between domain names and IP addresses. This means that almost every activity on the Internet starts with a DNS query (and often several) [1]. Along with the importance and evolvement of DNS, there is increasing concern over its vulnerability to attacks and failures. Consequently, multiple mitigation strategies have been raised to reduce the risk of a negative impact on all Internet-based resources [2].

Due to one of the main requirements on the DNS is its availability [3], a considerable amount of literature has been published on the topic of automatic DNS anomaly detection. These anomalies are commonly defined as traffic patterns that do not conform to expected normal behavior [4], [5], [6]. Identifying these unexpected DNS patterns is important, as detecting irregularities in the system's behavior could inform about abnormal or malicious events.

Additionally, as DNS is a distributed hierarchical database, top-level DNS servers are more likely to be the entry of DNS query behavior of clients [7]. Therefore, TLD name servers are more appropriate to be indicators of network anomalies since these anomalies can be reflected on their DNS traffic.

Hence, a system that fulfills the goal of automatic anomaly detection could be of great benefit for TLD registry operators, since offering its users a reliable DNS service is crucial for their business. Early detection of threats would allow them to take quick action, to preserve the system's correct functioning.

The problem of DNS anomaly detection is, however, a challenging task, as DNS traffic naturally evidences some well-known challenges for general anomaly detection methodologies [4]:

1) Defining a normal DNS traffic behavior that encompasses every possible normal behavior is very difficult. In addition, the boundary between normal and anomalous traffic behavior is not precise.
2) When DNS anomalies are the result of malicious actions (attacks), the adversaries could try to adapt themselves to make the anomalous DNS traffic appear normal.
3) DNS traffic behavior can frequently evolve due to the evolution of Internet data usage or changes in the system's configurations. Thus, a current notion of normal DNS traffic behavior might not be sufficiently representative in the future.
4) DNS traffic statistics measured at distinct scenarios can be significantly different from each other. Thus, applying a technique developed in one scenario to another is not straightforward.
5) There is a lack of labeled data for training/validation of DNS anomaly detection techniques.
6) DNS traffic data can contain random noise that tends to be similar to the actual traffic anomalies and hence is difficult to distinguish.

This paper presents a methodology for Anomaly Detection Based on Prediction (AD-BoP), by using a machine learning model to forecast different portions of the whole DNS traffic. The proposed method offers an effective mechanism to detect anomalies in the traffic of top-level domain name servers, as it was designed to face the six previously mentioned challenges successfully:

1) By using a learning approach, the method can be fed with a large amount of historical DNS data. Thus, the model defines a normal behavior from a large number of normal DNS traffic examples.
2) Even when DNS attacks can try to stay hidden when analyzing the overall DNS traffic, they will inevitably impact some portion of the traffic. Thus, by analyzing different portions of the traffic separately, AD-BoP enhances the detection of these malicious anomalies.
3) By using a learning approach, the AD-BoP method is

D. Madariaga, J. Madariaga, M. Panza and J. Bustos-Jiménez are with NIC Chile Research Labs, University of Chile.
E-mail: {diego, javier, jbustos}@niclabs.cl

D. Madariaga and B. Bustos are with the Millennium Institute for Foundational Research on Data, Department of Computer Science, University of Chile.

continuously updated and adapted to the evolution of DNS traffic behavior. Thus, the notion of normal DNS traffic is also being updated.

4) The presented anomaly detection strategy only depends on the behavior of the scenario where it is deployed. Therefore, AD-BoP can be applied to different DNS scenarios as it will learn the normal behavior in each case. This study shows the satisfactory performance of the AD-BoP method when applied to significantly different scenarios of real TLD name servers.
5) The lack of labeled data is not a problem for AD-BoP, since it uses an unsupervised learning approach. Consequently, AD-BoP is also able to detect new unknown anomalous events.
6) Other works that strictly focus on network attacks consider the detection of noisy traffic as a false positive [8]. By contrast, this study does not consider that the detection of these noisy traffics is incorrect per se, since they can be actually produced by other causes as misconfigurations or system failures.

For a better evaluation of the proposed AD-BoP method, this study compares its performance against two state-of-the-art methodologies for DNS anomaly detection in TLD name servers. As a result, AD-BoP is shown to present some improvements to the state-of-the-art methods:

- AD-BoP presents a simpler interpretation of the sensitivity parameter (threshold) selected.
- AD-BoP presents a direct interpretation of detected anomalies, which could help TLD operators rapidly find out the real cause of the abnormal behavior.
- AD-BoP presents a better performance at detecting some well-known DNS attacks, clearly separating those attacks from normal DNS behavior.

This paper presents an important contribution to the field of DNS anomaly detection, providing a useful and easily explainable methodology to detect DNS anomalies, based on the expected DNS traffic statistics. The proposed method could be especially helpful for TLD registry operators to preserve the reliability of their services.

II. Related Work

A considerable amount of literature has been published on detecting DNS anomalies using a wide variety of methodologies. Some research studies proposed methods to detect some specific DNS attacks, such as Domain Fluxing [9], Botnet Domains [10], and Kaminsky Cache Poisoning [11]. In this work, we propose an unsupervised learning approach to perform DNS anomaly detection, and therefore, our methodology is able to detect a wider variety of both known and unknown DNS anomalous events.

DNS anomaly detection can be roughly divided into two categories regarding where the DNS traffic is gathered. On the one hand, some works used probes installed in a particular network to gather DNS traffic, e.g., from university campus networks [12], [3]. On the other hand, some research studies analyzed the DNS traffic directly on DNS servers without additional monitoring infrastructure. The presented study uses the latter approach, by detecting DNS traffic anomalies in TLD name servers.

As TLD name servers are more likely to be indicators of network anomalies [7], a number of authors have reported traffic analyses for different TLD name servers. Wang *et al.* [7] used data from the Chinese .cn ccTLD to analyze the impact of a major national event on the DNS traffic behavior. In addition, Deri *et al.* [13] proposed a methodology for the Italian .it ccTLD for permanent DNS traffic monitoring. Researchers from InternetNZ, the registry for the .nz ccTLD for New Zealand, presented a time series approach to analyze trends on DNS queries and to inspect the presence of anomalies in historical DNS traffic [14]. Closer to the goals of this paper, Mikkle *et al.* [15] and Robberechts *et al.* [16] addressed the automatic anomaly detection for a TLD name server, using data from the .cz ccTLD for the Czech Republic and the .be ccTLD for Belgium, respectively. These two state-of-the-art anomaly detection methodologies, denominated as CZ.NIC method [15] and QLAD [16], serve as a baseline for our proposed approach.

Authors from CZ.NIC, the administrator of the .cz ccTLD for Czech Republic, implemented a profile-based anomaly detection methodology for extracting hidden anomalies in DNS traffic [15]. This method performs as follows:

1) All DNS packets inside a time window are represented by a tuple (t, A), where t is the arrival timestamp of the packet and A is a packet identifier. This identifier can be the source IP address or the first domain name in the DNS query. These two possible policies for A are referred to as source IP address policy and query name policy.
2) All A identifiers are hashed using N independent $k-$universal hash functions, with hash table size equal to M.
3) The method computes several sets denoted by $X_{n,m}$, containing all packets in which the n^{th} hash function mapped their A identifier to m.
4) Then, the sets $X_{n,m}$ are aggregated jointly over a collection of aggregation levels j with size J to form the $X^j_{n,m}(t)$ time series.
5) Each $X^j_{n,m}(t)$ is modelled using Gamma distribution laws with parameters α (shape) and β (scale). For every $X^j_{n,m}(t)$, α and β parameters are referred to as $(\alpha^j_{n,m}, \beta^j_{n,m})$.
6) For every hash function h_n, this method estimates the standard sample means with respect to m (α^j_n and β^j_n), and their variances ($\sigma^2_{n,\alpha,j}$ and $\sigma^2_{n,\beta,j}$).
7) Considering the Mahalanobis distance for $\gamma \in \{\alpha, \beta\}$ defined as

$$(D_{n,m})^2 = \frac{1}{J} \sum_{j=1}^{J} \frac{(\gamma^j_{n,m} - \gamma^j_n)^2}{\sigma^2_{n,\gamma,j}} \tag{1}$$

and considering the threshold value λ, if $D_{n,m} > \lambda$, then the set $X_{n,m}$ is said to contain at least one anomaly.
8) Finally, the intersection of all $X_{n,m}$ such that $D_{m,n} > \lambda$, corresponds to the detected anomalies within the scanned time window.

Several limitations to this method need to be acknowledged:

- One major drawback is that there is no clear interpretation of its threshold parameter, as it is applied after a complex data processing of the DNS traffic being analyzed.
- This method is supposed to fail to detect attacks that use randomly spoofed IP addresses because these malicious packets will belong to different sets [16]. This is an important limitation since the use of spoofed IP addresses is a widely used strategy at performing DNS attacks.

In addition, Robberechts *et al.* [16] used data belonging to the .be ccTLD for Belgium to present the implementation of their DNS anomaly detection system: QLAD. As their authors commented, it is a proof-of-concept system, which is divided in two subsystems named QLAD-global and QLAD-flow.

QLAD-global is a method based on the fact that DNS traffic anomalies will change the normal distribution of one or more traffic features. Thus, the detection of anomalies can be performed by detecting anomalies in the entropy of these traffic features. This method takes into account the following DNS traffic features:

- Number of DNS queries for each second-level domain name (1)
- Number of DNS queries for each record type (2)
- Number of DNS responses for each response code (3)
- Number of requests by each client (4), ASN (5), and country (6).
- DNS response sizes (7).

QLAD-global considers the division of the DNS traffic in successive windows of time T. At each time interval, the entropies of the traffic features are calculated by using the following equation:

$$H(X) = -\frac{1}{\log(n)} \sum_{i=1}^{n} p(x_i) \log(p(x_i)) \qquad (2)$$

where X is a traffic feature that can take values $x_1, ..., x_n$ with probability mass functions $p(x_1), ..., p(x_n)$. The division by $\log(n)$ (the maximum entropy) is adopted to avoid the effect of DNS traffic periodicity on the measured entropy, as proposed by Nychis *et al.* [17].

At each time window, the system computes the actual entropy of each traffic feature, denoted by e_t. Then, QLAD-global uses e_t to update an exponential moving average (EMA_t) and an exponential moving standard deviation (EMS_t). Finally, an anomaly is reported if the following is satisfied by any traffic feature:

$$\frac{|e_t - \text{EMA}_t|}{\text{EMS}_t} > \lambda \qquad (3)$$

where λ is the threshold that controls the sensitivity of the anomaly detection.

However, this method has a number of limitations for a real-world implementation:

- External databases must be maintained to perform geolocation of IP address. This is required to compute the number of requests by each ASN and country, whose information cannot be obtained directly from DNS packets.
- Threshold λ cannot be directly set, given the lack of prior knowledge about the expected entropy values. In addition, λ values do not have a direct interpretation regarding DNS traffic, hindering its understanding.
- The detection of an anomaly in the entropy of a traffic feature does not necessarily give enough information to a TLD operator to find out the real cause of the anomaly.

QLAD-flow method takes the previously mentioned work from CZ.NIC as a basis, with the addition of a new third policy named ASN policy, which is a generalization of the source IP address policy. However, in their experimental results, the ASN policy did not provide newer insights to the anomaly detection regarding the use of source IP policy. Therefore, only the QLAD-global subsystem of QLAD will be considered in this study, as QLAD-flow does not outperform the original CZ.NIC method.

III. AD-BoP Methodology

Our proposed Anomaly Detection Based on Prediction methodology (AD-BoP) is based on the fact that DNS anomalies will impact and change the expected traffic statistics for a given time interval. Thus, good predictions of these expected statistics can be used to detect anomalous traffic, according to the difference between real and expected values. Accordingly, this approach is well-aligned with the usual definition of an anomaly, defined as a pattern that does not conform to expected normal behavior [4], [5], [6].

In earlier research [18], we presented the feasibility of forecasting DNS traffic volume by using data from an authoritative name server for a ccTLD. We found that recurrent neural networks (specifically Long Short Term Memory networks) performed better than other simpler statistical models at predicting DNS traffic. The well-working of LSTM networks was reflected in their ability to capture the periodic patterns in the traffic and detect some abrupt phase changes.

Even though this previous work only studied the prediction of DNS traffic, it serves as the basis for developing our proposed anomaly detection method. In the present work, we take advantage of our previous findings regarding the feasibility of predicting DNS traffic to develop an anomaly detection mechanism based on the prediction of normal DNS traffic. By predicting DNS traffic for a future interval, we can use the difference between this prediction and the real traffic to indicate the presence of an anomaly.

As the volume of DNS traffic at a TLD is intrinsically very large, some traffic anomalies could affect just a particular segment of the traffic and stay hidden when analyzing the overall DNS traffic. Therefore, we are not just interested in predicting the whole volume of DNS traffic to find differences between predicted and real values. We are also interested in predicting some particular segments of DNS traffic, to analyze in more detail the presence of abnormal traffic behavior.

For that reason, we propose the aggregation of DNS data into different groups to make independent predictions. This approach could help TLD operators obtain more details about the specific sections of the whole DNS traffic affected by a given anomaly.

Given the aforementioned, our proposed AD-BoP method is focused on the following nine DNS traffic features:

- Number of DNS queries of types `A` (1), `AAAA` (2), `NS` (3), `MX` (4), and `ANY` (5).
- Number of unique queried domains (6).
- Number of DNS response packets with codes `NXDOMAIN` (7) and `NOERROR` (8).
- Total number of DNS packets (9).

AD-BoP considers a division of the DNS traffic in successive windows of time T. For each time interval, AD-BoP calculates a feature vector with the nine traffic features. Then, by consecutively joining these feature vectors, AD-BoP creates a multivariate time series, where each of its nine DNS-related features describes the behavior of a specific portion of the DNS traffic. Fig. 1 shows the temporal behavior of four of the nine selected features for a real authoritative name server for the Chilean .cl ccTLD. The figure contains two days of real data, where the differences between each time series are clearly visible.

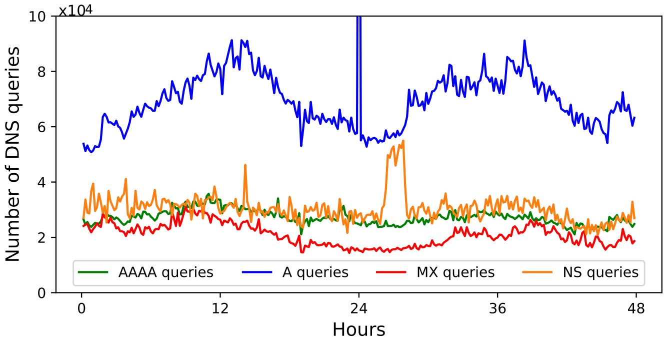


Fig. 1. Differences in the behavior of 4 DNS traffic features at a TLD name server

Furthermore, Fig. 1 illustrates a clear example of why the aggregation of DNS data into different groups can enhance the detection of anomalies, as some anomalies could impact only a specific segment of the DNS traffic. In the figure, around hour 24, there is a very pronounced anomaly in the number of DNS queries for `A` records (blue line), which is not visible at any of the other three time series. In addition, a second anomaly is visible around hour 26 with a duration of approximately three hours. This second anomaly only affects the number of DNS queries for `NS` records (orange line).

As suggested by our previous work [18], we used a forecast model based on LSTM networks to predict the time series regarding the nine selected DNS features. We modeled the problem as the prediction of one multivariate time series rather than the prediction of nine individual time series. Each DNS-feature depends not only on its past values but also on other DNS variables because, together, they all represent the behavior of DNS clients. Thus, we found a multivariate time series analysis more appropriate.

A. LSTM Forecast Model

Artificial Neural Network models, based on the natural neural networks of the human brain, consist of the interconnection of several nodes that individually define weights and operations to transmit data over their own connections and thus, over the network. This transmission of data allows the network to learn from the input to give predictions to future information.

Recurrent Neural Networks (RNNs) are a class of neural networks that include loops within their connections as a mechanism to store information in their cells and make it persist through the steps of the training process. In this way, the output obtained by each neuron is influenced by both the new input and the values obtained from previous computations. This characteristic is useful when dealing with time series or sequential data, and consequently, RNNs are widely used for time series prediction [19], [20].

LSTM is a particular type of recurrent neural network created to enhance the performance on Long-Term dependencies, as it deals better with the vanishing gradient problem related to the constant backpropagation process [21]. It achieves this by adding a new module to determine what information to store and forget.

The implementation of an LSTM unit consists of three gates: input gate, output gate, and forget gate that are related according to the following equations:

$$c_t = i_t \circ \tanh(W_c x_t + V_c y_{t-1} + b_c) + f_t \circ c_{t-1} \tag{4}$$

$$y_t = o_t \circ \tanh(c_t), \tag{5}$$

where i_t, f_t, o_t are the respective activation functions of each gate:

$$g_t = \sigma(W_g x_t + V_g y_{t-1} + b_g), \tag{6}$$

where σ is the sigmoid function, g is the corresponding gate, W and V are weight matrices, b is a bias vector, x_t and y_t are input and output vectors of the step t, and $\circ$ corresponds to the entry-wise product between two matrices.

B. Detection Strategy

After a given time interval $t-1$, AD-BoP employs the LSTM model to compute a prediction for the next interval t regarding the nine traffic features. Then, after the completion of interval t, AD-BoP obtains the nine real traffic features for interval t (from real DNS traffic), and uses them to update the LSTM model for next prediction. AD-BoP will label the interval t as anomalous if the following is satisfied by any of the nine traffic features:

$$\frac{|y_t - y'_t|}{y'_t} > \lambda \tag{7}$$

where y'_t denotes the predicted value and y_t denotes the real value of a given traffic feature at interval t. The λ value is the threshold that controls the method's sensitivity and is defined as the maximum accepted relative change of the traffic feature, calculated from the absolute difference between expected and real values with reference to the expected value.

λ value may not necessarily be equal for all traffic features, and it can be independently set for each one. However, in the present work, we will always consider the same λ value for the nine traffic features to enhance the readability of our findings.

Unlike threshold values used for controlling the sensitivity in both anomaly detection methods explained in Section II (CZ.NIC method and QLAD-global), our threshold parameter can be directly interpreted in terms of DNS traffic, as shown in the following example:

”The number of DNS queries for nonexistent domains is more than 60% higher than expected.”

This easy interpretation of detected anomalies could be very helpful for TLD operators to find out the real cause of the detected abnormal behavior.

IV. DNS Anomaly Injection

One of the main limitations of testing the effectiveness of DNS server monitoring and troubleshooting tools is the lack of labeled anomalies in the historical traffic data from real DNS servers. This means that DNS anomaly detection methodologies cannot be evaluated using common metrics such as precision and recall.

To evaluate the proposed anomaly detection methodology, a useful tool was designed to simulate anomalous traffic and to inject it into real DNS data (the source code is publicly available on https://github.com/niclabs/dns-anomaly-injection). This tool takes as input `.pcap` files with DNS traffic from a TLD authoritative name server, and returns new `.pcap` files with injected anomalies in addition to the original traffic. These artificially created DNS anomalies are customizable in terms of duration and magnitude of the attacks to provide different scenarios to test. The designed tool is able to simulate some well-known attacks to the DNS infrastructure:

Random Subdomain: In this attack, many DNS queries are sent to the DNS server for a single target domain. These queries are created by adding randomly generated subdomain to the victim’s domain. Random subdomain attacks cause the authoritative name servers of the target domain to experience DDoS, and responses may never come back from the target domain [22]. Even when the target of this attack is not the authoritative name server for the TLD, if the attack is distributed among many open resolvers, the evidence of the attack will also be visible on the TLD servers.

DNS Amplification: Like other amplification attacks, DNS Amplification is a type of reflection attack, where TLD nameservers are exploited as unknowing agents to perform the attacks [23]. In this case, the reflection is achieved by sending small DNS queries that result in large DNS responses to a spoofed IP address. DNS Amplification can be performed by using the EDNS0 DNS protocol extension, which allows for large DNS messages, or using the cryptographic feature of the DNS security extension (DNSSEC) to increase message size. In addition, this attack can be performed by using DNS record type `ANY`, which returns all records of all types known regarding the queried domain in a single request. The designed DNS anomaly injection tool implements this last type of DNS Amplification attack, which uses query type `ANY` to amplify DNS queries.

NXDOMAIN Flood: This is a type of DNS flood attack that attempts to overwhelm server resources and impact performance. It works by sending a flood of queries for nonexistent domain names to an authoritative name server. This attack causes the server’s cache to fill up with `NXDOMAIN` results, slowing DNS server response time for legitimate requests [24].

V. Data Set Overview

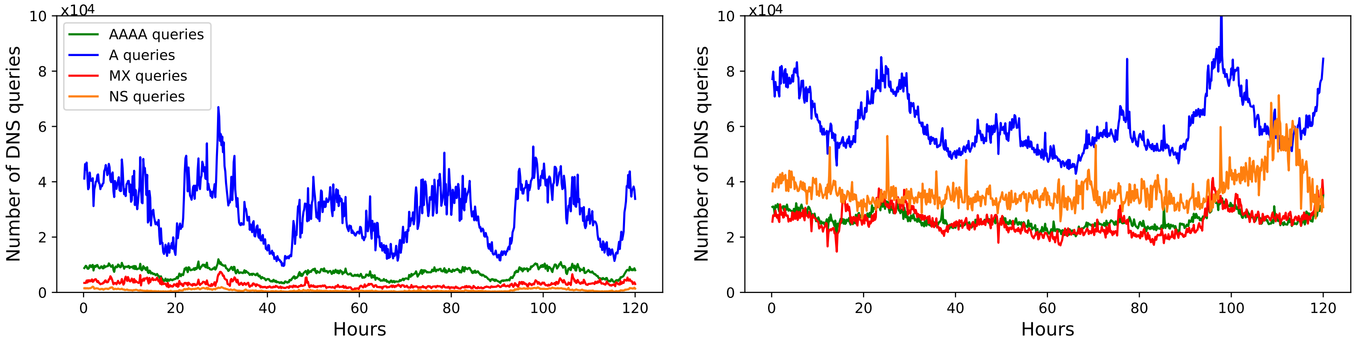
To test the proposed AD-BoP method in a real-world environment, we used data collected directly by the official registry for the Chilean .cl ccTLD: NIC Chile (Network Information Center of Chile) [25]. NIC Chile maintains a network of name servers for the .cl ccTLD worldwide to provide a robust and stable service with excellent response times. The data used in this study consist of a month of normal operation traffic from two authoritative name servers under the control of NIC Chile, belonging to an anycast configuration along with other servers [26]. The first name server is located in Santiago, Chile, whereas the second name server is located in Amsterdam, Netherlands. In the following, the collected data from these two servers will be referred to as *Santiago* dataset and *Amsterdam* dataset.

The collected data from both servers start on 7 November 2018, until 6 December of the same year. For each server, the dataset is divided into 4180 `.pcap` files, containing each one 10-min of DNS traffic data. This represents a total of 480 GB of raw `.pcap` files (110 GB for *Santiago* dataset and 370 GB for *Amsterdam* dataset).

These two DNS servers were selected as they represent different scenarios for real authoritative TLD name servers, having different traffic statistics. As indicated in Table I, *Santiago* server received an average of 80 queries per second (QPS), whereas *Amsterdam* server received an average of 263 QPS, that is, more than 3 times higher than *Santiago* server. In addition, 25% of DNS queries received by *Amsterdam* server were queries for nonexistent domains, in contrast with *Santiago* server, where just the 5% of the DNS queries were for nonexistent domains. Fig. 2 illustrates their differences in

TABLE I
Comparison of DNS traffic statistics between Santiago and Amsterdam datasets

	Santiago	**Amsterdam**
Average queries per second	80	263
% of DNS queries for nonexistent domains	5%	25%
% of DNS queries of type `A` (IPv4 address record)	66%	40%
% of DNS queries of type `AAAA` (IPv6 address record)	16%	17%
% of DNS queries of type `MX` (Mail exchanger record)	7%	14%
% of DNS queries of type `NS` (Name server record)	2%	22%



(a) Santiago dataset

(b) Amsterdam dataset

Fig. 2. Comparison of different DNS traffic features between Santiago and Amsterdam datasets.

the number of DNS queries for four of the most requested record types: `AAAA`, `A`, `MX` and `NS`. *Santiago* server mostly received DNS requests for `A` records (66%), and presents low percentages of requests for `MX` records (7%) and `NS` records (2%). On the other hand, *Amsterdam* server received a lower percentage of requests for `A` records (40%), but higher percentages of requests for `MX` records (14%) and `NS` records (22%).

As it is well known, in the problem of anomaly detection in network traffic (and particularly in DNS traffic), the false-negative rate cannot be obtained since there is no prior certainty about the presence or absence of anomalies at each data point. Thus, the relevance of DNS anomaly detection methodologies cannot be evaluated using common measures as recall.

Therefore, to test the AD-BoP method and compare it against other state-of-the-art methodologies, some artificially created anomalies were injected inside the DNS traffic of both *Santiago* and *Amsterdam* datasets. Thereby, we generated a minimal set of data points that a DNS anomaly detector would be expected to detect. It is important to underline that this latter does not imply that the selection of these injected anomalies was influenced by our proposed method. Indeed, in the following, we provide an exhaustive explanation about why a DNS anomaly detector is expected to detect these traffic injections since they correspond to real-world anomalies that could (1) compromise the well-working of a TLD server or (2) use the normal operation of the DNS infrastructure as a weapon against a targeted victim's server.

By using the tool mentioned in Section IV, 9 different DNS anomalies were injected into the last week of both datasets (last 1,008 `.pcap` files with 10-min traffic intervals). Each anomaly was injected inside a different 10-min traffic interval, and therefore, 9 out of the 1,008 `.pcap` files were labeled as certainly anomalous. In both *Santiago* and *Amsterdam* datasets, the 9 injected anomalies corresponded to 3 random subdomain attacks, 3 DNS amplification attacks, and 3 NXDOMAIN flood attacks. The injected attacks were varied in terms of magnitude, and they all correspond to DNS anomalies that an anomaly detection system should be able to detect, as they could (1) compromise the well-working of a TLD server (NXDOMAIN flood attacks) or (2) use the normal operation of the DNS infrastructure as a weapon against a targeted victim's server (DNS amplification and random subdomain attacks).

Table II illustrates the magnitude of the injected anomalies in both *Santiago* and *Amsterdam* datasets. In particular, each anomaly is labeled with its duration (ΔT) and with the average queries per second generated by the anomaly (QPS). Thus, each anomaly injected $\left(\text{QPS} \times \Delta\text{T}\right)$ DNS packets into the traffic received by the TLD servers.

TABLE II
MAGNITUDE OF INJECTED ANOMALIES IN BOTH SANTIAGO AND AMSTERDAM DATASETS

	Santiago		Amsterdam	
	ΔT [s]	Avg. QPS	ΔT [s]	Avg. QPS
Random subdomain #1	30	100	30	100
Random subdomain #2	30	1,000	30	1,000
Random subdomain #3	30	10,000	30	10,000
DNS amplification #1	30	10	30	10
DNS amplification #2	60	10	60	10
DNS amplification #3	120	10	120	10
NXDOMAIN flood #1	30	100	30	600
NXDOMAIN flood #2	30	1,000	30	1,200
NXDOMAIN flood #3	30	10,000	30	2,400

As it can be seen, NXDOMAIN flood attacks increase the DNS traffic in at least one order of magnitude from the average queries per second normally received by both servers (Table I). These magnitudes are consistent with the fact that NXDOMAIN flood attacks attempt to compromise the well-working of the TLD server receiving the queries (*Santiago* and *Amsterdam* servers). For random subdomain and DNS amplification, the TLD servers are not the target of the DoS attacks. Indeed, the real targeted victim's server could receive traffic from multiple TLD simultaneously, and therefore, just a minor portion of the entire attack's traffic will be visible inside *Santiago* and *Amsterdam* servers' traffic. Accordingly, the different magnitudes of these two types of DNS attacks are coherent with reality. On the one hand, random subdomain attacks cause DoS by generating many DNS queries to the targeted server. On the other hand, DNS amplification attacks cause DoS by generating large DNS responses (in bytes) to the targeted IP address. Thus, random subdomain attacks are more dependent on high QPS values than DNS amplification attacks, and therefore, DNS amplification anomalies were configured with lower QPS.

VI. Experimental Results

This section presents the experimental results at testing the proposed DNS anomaly detection system (AD-BoP) in addition to the other two state-of-the-art methods (CZ.NIC method and QLAD-global). The three methodologies were tested on the last week of both *Santiago* and *Amsterdam* datasets, represented by 1,008 consecutive `.pcap` files, each one with 10 minutes of DNS traffic data. As mentioned before, both datasets contain nine artificially created anomalies among its 1,008 `.pcap` files, representing the 0.9% of the 1,008 traffic windows. Thus, at each of these 1,008 time intervals, the methods are required to determine the presence or absence of anomalous traffic behavior.

As indicated in Sections II and III, the three methodologies employ a sensitivity (or threshold) parameter to determine the presence of anomalies. Hence, an exhaustive threshold analysis was performed to evaluate the methods being compared at different configurations. In this analysis, several sensitivity values were tested, quantifying the total number of 10-min intervals labeled as anomalies (from 0 to 1,008) and the number of intervals with injected attacks correctly labeled as anomalies (from 0 to 9).

It is important to emphasize that the nine artificially injected DNS attacks corresponded to anomalies on a TLD server's normal operation that an anomaly detection system would be expected to detect. Thus, it is also relevant to identify the threshold values which allow the detection of all these nine attacks.

In the case of the AD-BoP method, it needed to create and train its LSTM model before analyzing the last week of both datasets. Thus, for each *Santiago* and *Amsterdam* servers, AD-BoP firstly created a forecast model using the first three weeks of *Santiago* and *Amsterdam* datasets, respectively. These three weeks corresponded to the first 3,172 `.pcap` files of both datasets, preceding the 1,008 time intervals used for testing the anomaly detection methods. The 3,172 time intervals were split into train and validation sets using a 75% / 25% ratio to create LSTM forecast models according to the following configuration:

- **Training:** The model was fed with batches of size 24 and trained for 30 epochs using *Adam* optimizer and mean absolute error (MAE) as the loss function.
- **Hidden Layers:** An LSTM layer of size 150 follows with a dropout value of 0.3 and using hyperbolic tangent as the activation function.
- **Input:** The model receives as input one week of data, that is to say, 1,008 multivariate values corresponding to 1,008 10-min traffic intervals.
- **Output:** The output of the model is a unique 9-dimensional value, which corresponds to the prediction of the DNS features for the next 10-min window. This value is used to retrain the model, updating its weight values before performing the next prediction.

With respect to the configuration of CZ.NIC method, its default and recommended parameters were used: 8 aggregation levels, 25 hash functions, and hash tables of size 32.

A. Santiago dataset

Fig. 3 shows the result of applying AD-BoP methodology over the last week of *Santiago* dataset using different threshold configurations. The red line shows the total number of 10-min windows labeled as anomalies (from 0 to 1,008) as a function of the threshold value. Similarly, the blue line shows the number of injected attacks correctly labeled as anomalies (from 0 to 9) as a function of the threshold value.

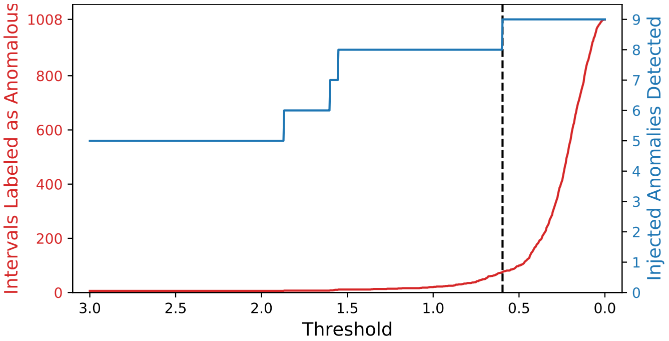


Fig. 3. Anomaly detection in *Santiago* dataset using AD-BoP method. In red: total anomalies detected in function of threshold value. In blue: injected anomalies detected in function of threshold value.

According to Fig. 3, the threshold value needed to detect the nine injected anomalies is 0.684. This means that there is a difference of more than 68.4% between expected and real values for at least one traffic feature in the nine traffic windows with injected anomalies. At this threshold configuration, AD-BoP classified a total of 59 10-min windows as anomalous, out of a maximum of 1,008 traffic windows. That is, 5.9% of the whole *Santiago* dataset was labeled as anomalous behavior using this threshold.

When analyzing the number of total traffic windows labeled as anomalous in Fig. 3 (red line), this curve is naturally separated into two different zones. Firstly, from a threshold value of 3.0 to approximately 0.5, there is a slow increase in the number of traffic windows labeled as anomalous. Secondly, from a threshold value of 0.5 to 0, there is a faster growth of the curve, rapidly reaching the detection rate of 100% (all 1,008 intervals labeled as anomalous). With respect to the nine injected anomalies, they all were detected in the first slow-growing zone, being clearly separated from the vast majority of traffic intervals.

Fig. 4 shows three of the nine DNS-related time series used by the AD-BoP method when analyzing the 1,008 10-min windows from *Santiago* dataset. These time series correspond to 1) the number of distinct queried domains, 2) the number of DNS queries for `ANY` records, and 3) the number of DNS responses with response code `NXDOMAIN`. In addition, vertical red lines identify the position of the nine injected DNS attacks. This figure exemplifies one of the main claims on which this method is based: DNS anomalies can affect just a particular portion of the whole traffic. Hence, anomalies marked as

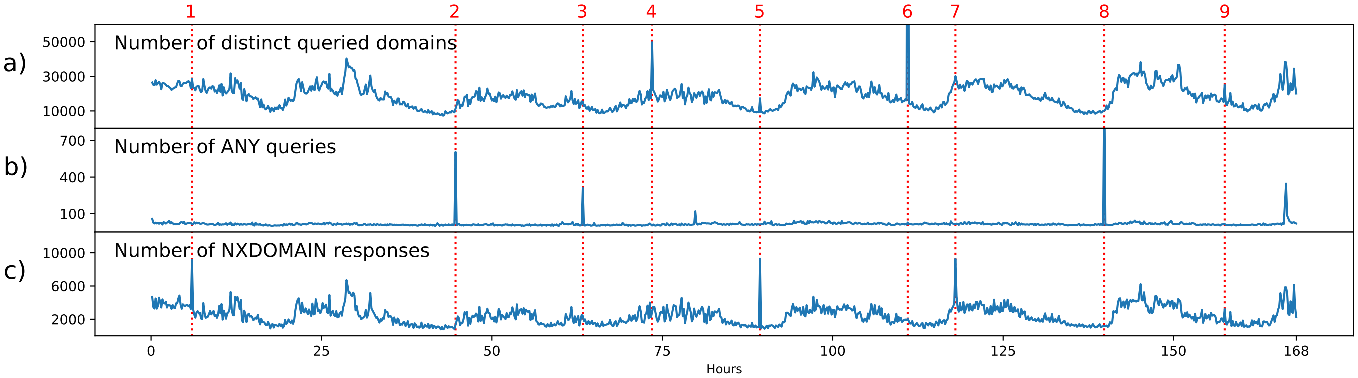


Fig. 4. Time series for the last 1,008 10-min windows in *Santiago* dataset, regarding a) the number of distinct queried domains, b) the number of `ANY` queries and c) the number of responses for `NXDOMAIN`. In addition, the nine injected anomalies are identified with red vertical lines.

1, 5, and 7 correspond to DNS `NXDOMAIN` floods, which significantly impact the normal amount of DNS responses with code `NXDOMAIN` (Fig. 4.c). Anomalies marked as 2, 3, and 8 correspond to DNS amplification attacks, and they significantly impact the normal traffic of DNS queries for `ANY` records (Fig. 4.b). Lastly, anomalies marked as 4, 6, and 9 correspond to random subdomain attacks, significantly impacting the normal number of distinct requested domains (Fig. 4.a).

Furthermore, a visual inspection of the time series in Fig. 4 reveals the abrupt changes in normal traffic behavior induced by the injected attacks. This supports the idea of why DNS anomaly detection systems would be expected to detect these anomalies.

In addition, CZ.NIC method was applied over the last week of *Santiago* dataset using its two possible policies: source IP address policy and query name policy. As both policies use their own threshold configuration, they were analyzed separately, as shown in Fig. 5. Using the source IP address policy, this method identified the nine injected anomalies at a threshold value of 1.199. At this configuration, CZ.NIC method classified 975 10-min windows as anomalous, i.e., 96.7% of the whole dataset. On the other hand, the query name policy identified all injected anomalies with a threshold value of 1.779, classifying a total of 875 10-min windows as anomalous, i.e., 86.8% of all 10-min windows.

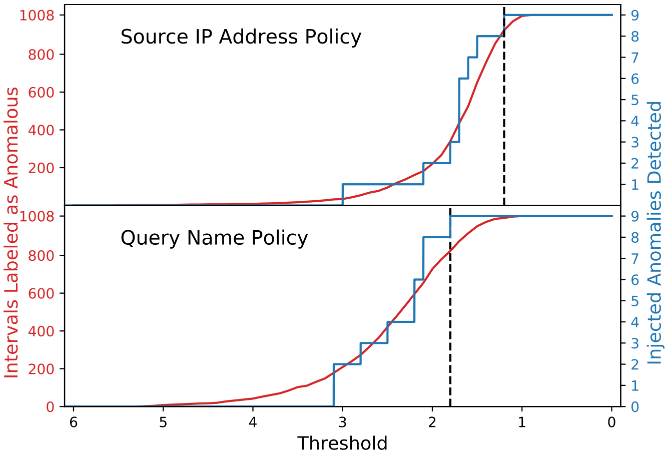


Fig. 5. Anomaly detection in *Santiago* dataset using CZ.NIC method. The figure shows the behavior when using source IP address policy, and query name policy. In red: total anomalies detected in function of threshold value. In blue: injected anomalies detected in function of threshold value.

When analyzing the number of total traffic windows labeled as anomalous in Fig. 5 (red lines), the curves for both policies present similar behaviors. The curves are naturally separated into two different zones: a first zone of slow growth, followed by a zone of faster growth. Differently from AD-BoP results, the nine injected anomalies were not detected in the first slow-growing zones. Moreover, the number of injected anomalies detected (blue lines) presents a similar behavior to the total number of intervals labeled as anomalous (red lines). Then, the detection of all these nine anomalies occurs at a high detection rate (more than 85% of all 1,008 intervals labeled as anomalous).

Since both policies were designed to complement each other, the behavior of CZ.NIC method using both policies at the same time was also studied. Therefore, all the combinations of both thresholds that allow the detection of the 9 injected attacks were analyzed. The most efficient configuration to detect all these injected anomalies was found for both threshold values of 2.0. At this configuration, CZ.NIC method classified a total of 779 10-min windows as anomalous, i.e., 77.3% of all traffic windows.

With respect to QLAD-global, Fig. 6 shows the result of applying this methodology over the last week of *Santiago* dataset using different threshold configurations. This method identified the nine injected anomalies at a threshold value of 1.224. At this sensitivity, QLAD-global labeled 548 10-min windows as anomalous. That is, 54.4% of all the 1,008 possible windows.

When analyzing the number of total traffic windows labeled as anomalous in Fig. 6 (red line), the curve starts with a fast-growing behavior, rapidly reaching high detection rate values. This implies that, even when the detection of all the nine anomalies occurs rapidly, it appears with a high detection rate (more than 50% of all 1,008 intervals labeled as anomalous).

Finally, as QLAD-global was designed to complement CZ.NIC method (referred to as QLAD-flow in [16]), QLAD-

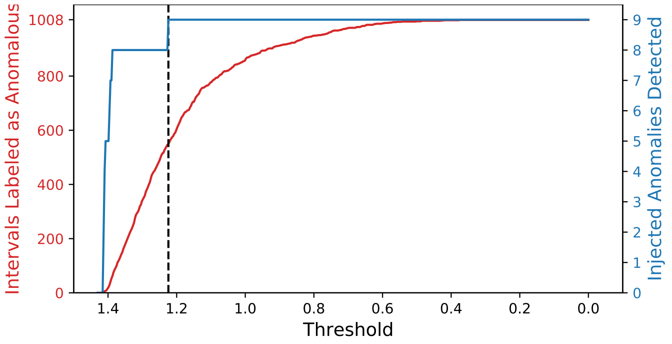


Fig. 6. Anomaly detection in *Santiago* dataset using QLAD-global. In red: total anomalies detected in function of threshold value. In blue: injected anomalies detected in function of threshold value.

global and CZ.NIC method (using both policies) were analyzed together. Therefore, we analyzed all the combinations of thresholds that allow the detection of the nine injected attacks. The most efficient configuration to detect all these injected anomalies was found for QLAD-global's threshold of 1.387, query name policy's threshold of 2.4, and not considering source IP address policy. At this configuration, 511 traffic windows were classified as anomalous, i.e., 50.7% of the whole dataset.

B. Amsterdam dataset

For the *Amsterdam* dataset, the same experiments as for the *Santiago* dataset were performed.

Fig. 7 shows the result of applying AD-BoP methodology over the last week of *Amsterdam* dataset using different threshold configurations. The threshold value needed to detect the nine injected anomalies is 0.348. This means that there is a difference of more than 34.8% between expected and real values for at least one traffic feature in the nine intervals with injected anomalies. At this threshold configuration, AD-BoP classified a total of 95 10-min windows as anomalous, out of a maximum of 1,008 traffic windows. That is, 9.4% of the whole *Amsterdam* dataset was labeled as anomalous by using this threshold.

When analyzing the number of total traffic windows labeled as anomalous in Fig. 7 (red line), this curve is naturally separated into two different zones, in the same way as for the *Santiago* dataset. Firstly, from a threshold value of 3.0 to approximately 0.2, there is a slow increase in the number of traffic windows labeled as anomalous. Secondly, from a threshold value of 0.2 to 0, there is a faster growth of the curve, rapidly reaching the detection rate of 100% (all 1,008 intervals labeled as anomalous). As for the *Santiago* dataset, the nine injected anomalies were all detected in the first slow-growing zone, being clearly separated from the vast majority of traffic intervals.

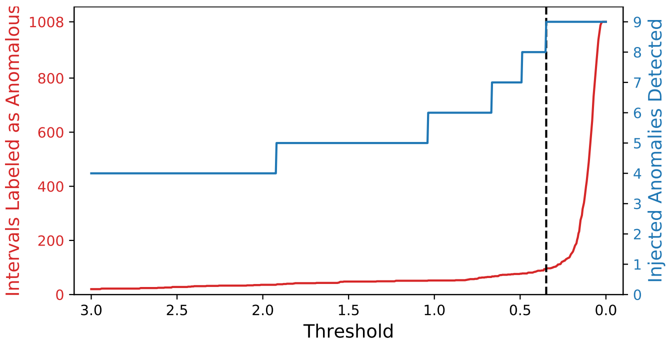


Fig. 7. Anomaly detection in *Amsterdam* dataset using AD-BoP method. In red: total anomalies detected in function of threshold value. In blue: injected anomalies detected in function of threshold value.

Similarly to *Santiago* dataset, Fig. 8 shows three of the nine DNS-related time series used by the AD-BoP method when analyzing the 1,008 traffic windows from *Amsterdam* dataset. These time series correspond to 1) the number of distinct queried domains, 2) the number of DNS queries for `ANY` records, and 3) the number of DNS responses with response code `NXDOMAIN`. In addition, vertical red lines identify the position of the nine injected DNS attacks. As for the *Santiago* case, this figure exemplifies the fact that DNS anomalies can affect just a particular segment of the whole traffic. Anomalies marked as 1, 5, and 7 correspond to DNS `NXDOMAIN` floods, which significantly impact the normal amount of DNS responses with code `NXDOMAIN` (Fig. 8.c). Anomalies marked as 2, 3, and 8 correspond to DNS amplification attacks, and they significantly impact the normal traffic of DNS queries for `ANY` records (Fig. 8.b). Lastly, anomalies marked as 4,

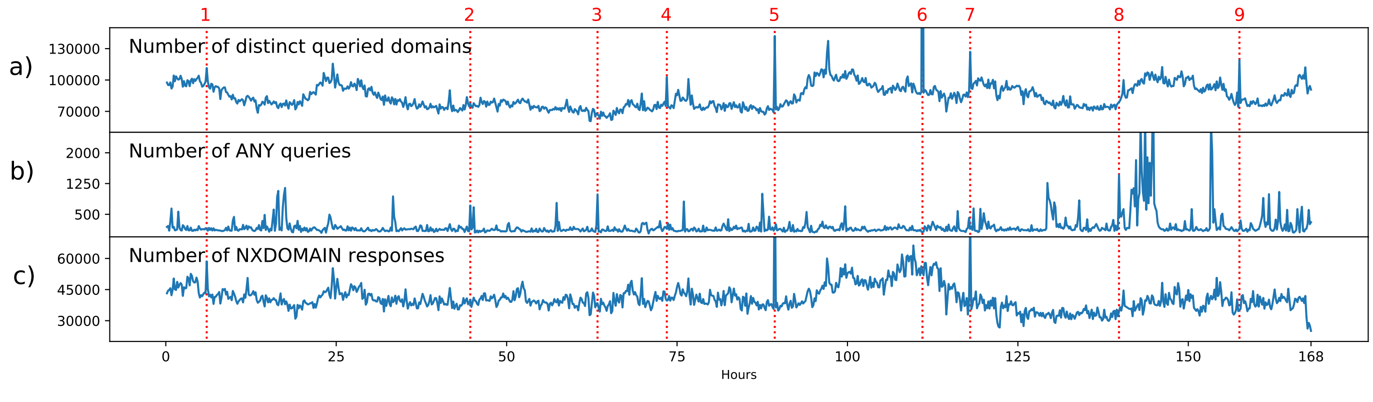


Fig. 8. Time series for the last 1,008 10-min windows in *Amsterdam* dataset, regarding a) the number of distinct queried domains, b) the number of `ANY` queries and c) the number of responses for `NXDOMAIN`. In addition, the nine injected anomalies are identified with red vertical lines.

6, and 9 correspond to random subdomain attacks, which significantly impact the normal amount of distinct requested domains (Fig. 8.a).

Regarding CZ.NIC method, it was applied over the last week of *Amsterdam* dataset using its two possible policies with different threshold configurations, as shown in Fig. 9. Using the source IP address policy, this method identified the nine injected anomalies at a threshold value of 1.399. At this configuration, CZ.NIC method classified 1,006 10-min windows as anomalous, i.e., 99.8% of the whole dataset. On the other hand, the query name policy identified all injected anomalies with a threshold value of 0.699, classifying a total of 1,008 10-min windows as anomalous, i.e., 100% of all 10-min windows.

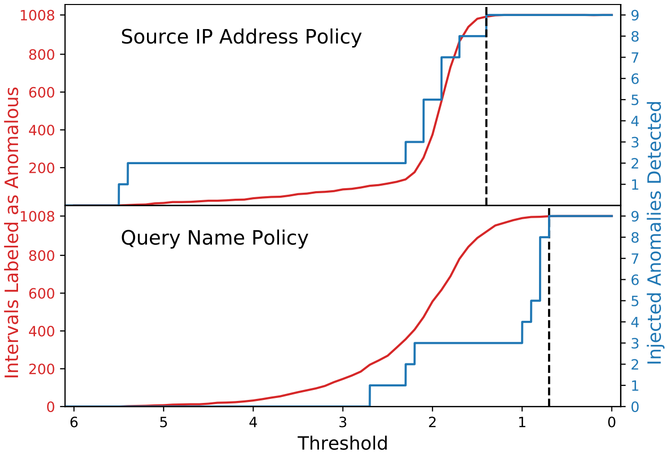


Fig. 9. Anomaly detection in *Amsterdam* dataset using CZ.NIC method. The figure shows the behavior when using source IP address policy, and query name policy. In red: total anomalies detected in function of threshold value. In blue: injected anomalies detected in function of threshold value.

When analyzing the number of total traffic windows labeled as anomalous in Fig. 9 (red lines), the curves for both policies present similar behaviors. As for the *Santiago* dataset, the curves are naturally separated into two different zones: a first zone of slow growth, followed by a zone of faster growth. The nine injected anomalies were not detected in the first slow-growing zones, replicating the behavior obtained in *Santiago* dataset. Then, the detection of all these nine anomalies occurs at a high detection rate (more than 90% of all 1,008 intervals labeled as anomalous).

In addition, when using both policies simultaneously, the most efficient configuration to detect all the injected anomalies was found, in this case, for only taking into account the source IP address policy.

With respect to QLAD-global, Fig. 10 shows the result of applying this methodology over the last week of *Amsterdam* dataset using different threshold configurations. This method identified the nine injected anomalies at a threshold value of 1.161. At this sensitivity, QLAD-global labeled 773 10-min windows as anomalous. That is, 76.7% of all the 1,008 possible traffic windows.

When analyzing the number of total traffic windows labeled as anomalous in Fig. 10 (red line), the curve starts with a fast-growing behavior, rapidly reaching high detection rate values. As for the *Santiago* dataset, this implies that, even when the detection of all the nine anomalies occurs rapidly, it appears with a high detection rate (more than 75% of all 1,008 intervals labeled as anomalous).

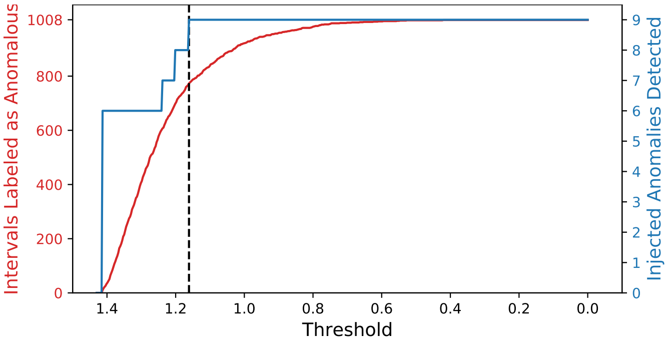


Fig. 10. Anomaly detection in *Amsterdam* dataset using QLAD-global method. In red: total anomalies detected in function of threshold value. In blue: injected anomalies detected in function of threshold value.

Finally, we also analyzed QLAD-global and CZ.NIC method (using its two policies) together. Thereby, the most efficient configuration to detect all the injected anomalies was found for QLAD-global's threshold of 1.409, source IP address policy's threshold of 5.3, and query name policy's threshold of 2.6. At this configuration, 262 traffic windows were classified as anomalous, i.e., 26.0% of the whole dataset.

Table III summarizes, for each of the compared methodologies, the percentage of 10-min windows labeled as anomalous when detecting the nine injected anomalies.

TABLE III
PERCENTAGE OF TRAFFIC WINDOWS LABELED AS ANOMALOUS AT THE FIRST DETECTION OF THE NINE INJECTED ANOMALIES.

	Santiago	**Amsterdam**
CZ.NIC source IP policy	96.7%	99.8%
CZ.NIC query name policy	86.8%	99.9%
CZ.NIC both policies	77.3%	99.8%
QLAD-global	54.4%	77.7%
QLAD-global + CZ.NIC both policies	50.7%	26.0%
AD-BoP	**5.9%**	**9.4%**

VII. DISCUSSION

A. Evaluation of Detected Anomalies

As mentioned in Section VI, the performance of the anomaly detection methodologies was analyzed based on the detection of a minimal set of artificially created DNS attacks. These nine injected attacks corresponded to anomalies that a successful DNS anomaly detector would be expected to detect. Consequently, we studied the threshold values that allow the methods to detect all the nine injected anomalies. Thus, we quantified the total number of traffic windows labeled as anomalous when detecting all these DNS attacks. This quantification can be used to measure the quality of each method, since if a method requires a very low threshold to detect the injected anomalies, it will classify a large number of normal traffic windows as anomalous, increasing the number of false positives.

According to Table III, CZ.NIC method labeled in all cases more than 75% of the whole dataset as anomalous when detecting the nine DNS attacks, even when applying both policies simultaneously. Similarly, QLAD-global also presents a high percentage of 10-min windows labeled as anomalies: 54.4% for *Santiago* dataset and 77.7% for *Amsterdam* dataset.

It is important to notice that these large numbers of traffic windows classified as anomalies are not incorrect per se, since there are some cases where anomalous traffic can be actually more frequent than normal traffic [5], e.g., when dealing with bursty DDoS attacks. However, when analyzing both *Santiago* and *Amsterdam* datasets with DNS experts from NIC Chile, we did not find situations similar to this. Therefore, the total number of real anomalies should not be as high as for QLAD-global and CZ.NIC methods. This claim is well aligned with the reasonable and widely accepted assumption that the majority of the network connections are normal traffic, vastly outnumbering the number of anomalous traffic [27].

Joining QLAD-global with CZ.NIC method using its two policies (as proposed by Robberechts *et al.* [16]) enhanced the effectiveness of detecting the injected DNS attacks. That is, the number of total traffic windows labeled as anomalies was reduced to 50.7% for *Santiago* dataset and to 26.0% for *Amsterdam* dataset. This decrease is especially important for *Amsterdam* dataset, as when applied separately, all the methods classified more than 75% of the whole dataset as anomalous. Nonetheless, this approach still incurs a high number of traffic intervals labeled as anomalies for *Santiago* dataset (50.7% of the whole dataset). Furthermore, this ensemble method is based on the proper selection of three different thresholds (as discussed in Section VI), which can be a complicated task to be properly achieved in practice.

Differently from the other approaches, the AD-BoP method presented a lower percentage of 10-min windows labeled as anomalies when detecting the nine injected attacks: 5.9% for *Santiago* dataset and 9.4% for *Amsterdam* dataset. These results are more coherent with the widely accepted assumption of a low percentage of anomalous network traffic [27].

According to these results, AD-BoP classified as anomalous 59 traffic intervals from *Santiago* dataset, where only nine corresponded to the injected anomalies. After inspecting the remaining 50 detected anomalies along with DNS experts from NIC Chile, it can be concluded that:

- 14 detected anomalies corresponded to heavy spamming/email marketing. These traffic anomalies came mostly from 10 different IP addresses, where 4 of them were listed in spam blacklists.
- 21 detected anomalies corresponded to DNS enumeration activities, including standard record enumeration (`A`, `NS`, `SOA`, and `MX` records) and `SRV` enumeration. These traffic anomalies came mostly from 14 different IP addresses, where 10 of them were found in IP blacklists.
- 9 detected anomalies corresponded to the query behavior of caching resolvers.
- 2 detected anomalies corresponded to random subdomain attacks. The IP address performing these anomalies was found in at least 5 different IP blacklists.
- 1 detected anomaly corresponded to a DNS amplification attack using spoofed queries of type `ANY`.
- The remaining 4 detected anomalies could not be clearly identified with any real anomalous behavior and were considered as false positives.

Therefore, when detecting the nine injected anomalies, AD-BoP incurred a false-positive rate of 6.8%.

In the same way, AD-BoP classified as anomalous 95 traffic intervals from *Amsterdam* dataset, where only nine corresponded to the injected anomalies. After inspecting the remaining 86 detected anomalies along with DNS experts from NIC Chile, it can be concluded that:

- 56 detected anomalies corresponded to subdomain enumeration using `ANY` records. The presence of these anomalies is clearly visible in Fig. 8.b, where the number of `ANY` queries is highly affected by these anomalies, especially at the end of the time series.
- 4 detected anomalies corresponded to heavy spamming/email marketing, with a highly distributed pattern.
- 12 detected anomalies corresponded to DNS enumeration activities using standard record enumeration (`A`, `NS`, `SOA`, and `MX` records). These traffic anomalies came mostly from 22 different IP addresses, where 10 of them were found in IP blacklists.
- 7 detected anomalies corresponded to DNS amplification attacks using spoofed queries of type `ANY`.
- The remaining 7 detected anomalies could not be clearly identified with any real anomalous behavior and were considered as false positives.

Therefore, when detecting the nine injected anomalies, AD-BoP incurred a false-positive rate of 7.4%.

When analyzing the results obtained by AD-BoP in both datasets, it can be shown that this method is able to detect a wide range of other real DNS anomalies besides the artificially created, incurring false-positive rates lower than 8%.

B. *Threshold Evaluation*

As the three methods being compared employ a threshold value to adjust their detection sensitivities, an exhaustive threshold analysis was performed in Section VI to evaluate the implications of different threshold values.

Considering that these methods are intended to be used by TLD registry operators, they will need to understand the meaning of the threshold value. A proper interpretation of the threshold value could be crucial in selecting a proper sensitivity configuration for their services.

Regarding the AD-BoP method, the interpretation of its threshold is direct. It represents the maximum accepted difference between expected and real values of any DNS feature with reference to the expected value. This easily explainable threshold value leads to easy interpretability of detected anomalies. For example, if the threshold is set to 0.5, a detected anomaly can be explained as a difference of more than 50% between the real and the expected value for one of the DNS-related features (number of queries for `A` records, number of distinct requested domains, number of queries for nonexistent domains, etc.).

With respect to CZ.NIC method, the interpretation of its threshold value is more complicated since it does not have a simple relation with the DNS traffic being analyzed. This is because its threshold is applied after a complex data processing of the DNS data, comprehending the use of different hash functions, gamma distribution fitting, the use of Mahalanobis distance, among other tasks. Consequently, configuring the sensitivity of CZ.NIC method is difficult since there is no clear relationship between a given threshold value and the expected behavior of the anomaly detection system.

In the case of QLAD-global, its threshold value can be roughly interpreted as a maximum accepted difference between the expected and real entropy values of the selected DNS features. As the threshold refers to the normalized entropy of the DNS features being analyzed, there is no preconception about the impact of possible anomalies on normalized entropies' values. Therefore, it is complicated to select a meaningful threshold value to be used in real scenarios. Also, when analyzing Fig. 6 and Fig. 10, the number of total detected anomalies rapidly increases when selecting threshold values lower than 1.4. This behavior makes it difficult to adjust a proper threshold value since very small variations can lead to the inclusion of a large number of false positives.

C. Performance Evaluation

In this subsection, the performance in terms of execution time is analyzed for the three anomaly detection methods being compared. Accordingly, the average time needed to label a 10-min traffic window as normal or anomalous was reported for each method. In the following, all values refer to the use of a 1.6GHz quad-core processor (Intel Core i5-8250U) with 8GB RAM.

CZ.NIC method required the shortest time to process a single 10-min `.pcap` file and classify it as normal or anomalous. When using its source IP address policy, CZ.NIC method required an average time of 0.24 s for *Santiago* dataset and 0.36 s for *Amsterdam dataset*. On the other hand, when using its query name policy, it required an average time of 0.94 s for *Santiago* dataset and 1.53 s for *Amsterdam* dataset.

Regarding QLAD-global, it required an average time of 10.32 s for *Santiago* dataset and 36.84 s for *Amsterdam* dataset. These values are considerably higher than the presented by Robberechts *et al.* [16], as QLAD's authors explicitly did not count the time needed to extract the traffic feature distributions from the `.pcap` files. However, we did count this time as the extraction of features from log files cannot be performed *offline*, and it must be completed before deciding the presence or absence of an anomaly.

With respect to the proposed AD-BoP method, a more detailed analysis is performed. When AD-BoP receives a new `.pcap` file for a time interval t, it performs as follows:

i) It analyzes the `.pcap` file to extract a vector $\hat{y}_t$ with the value of the nine DNS-related features for interval t.
ii) It compares these real values in vector $\hat{y}_t$ with the vector of predicted values $\hat{y}'_t$ (calculated in the previous interval $t-1$). According to the value of threshold λ, it decides if the interval t presents an anomaly or not.
iii) It updates its LSTM network model with real values in vector $\hat{y}_t$.
iv) It performs a one-step-ahead prediction and obtains a vector $\hat{y}'_{t+1}$ with the nine predicted traffic features for the next interval $t+1$.

Accordingly, to perform anomaly detection on a given 10-min traffic interval, only steps i and ii are necessary. Steps iii and iv must be completed before the next time interval, but they are not needed to decide the presence or absence of anomalies in the current interval.

Hence, to label a 10-min window (steps i and ii), AD-BoP required an average time of 7.9 s for *Santiago* dataset and 28.3 s for *Amsterdam* dataset. Steps iii and iv (which are not needed to label the current time window) were performed in an average time of 3.8 s for both *Santiago* and *Amsterdam* dataset.

Table IV summarizes the average time needed by each method to classify a 10-min traffic interval as normal or anomalous. Thereby, the AD-BoP method is shown to be competitive to the other methodologies in terms of execution time, considering its outstanding performance at correctly detecting the injected anomalies.

TABLE IV
AVERAGE TIME NEEDED TO DECIDE THE PRESENCE OR ABSENCE OF AN ANOMALY FOR A 10-MIN TRAFFIC WINDOW

	Santiago	**Amsterdam**
CZ.NIC source IP policy	0.24 s	0.36 s
CZ.NIC query name policy	0.94 s	1.53 s
QLAD-global	10.32 s	36.84 s
AD-BoP	**7.90 s**	**28.30 s**

D. Time Series Forecasting Method

As mentioned in Section III, the fundamental basis for the well-working of AD-BoP is its prediction step. To this end, we selected an LSTM model to forecast the DNS-related time series. This selection was based on its suitability for predicting DNS traffic, capturing the periodic patterns in the traffic, and detecting some abrupt phase changes [18]. However, the prediction step in AD-BoP methodology can be carried out by using other time series techniques. For example, methods more light-weighted than LSTM networks can be considered. In the following, we present a comparison of AD-BoP's results by using the LSTM model and Holt-Winters [28] as time series forecasting methods. Both methods are compared in terms of their effectiveness in detecting the injected anomalies and their execution time.

As in Section VII-A, the performance of both versions of the AD-BoP method was studied based on the detection of the nine injected attacks. Therefore, we analyzed the total number of traffic windows labeled as anomalous when detecting all the artificial DNS attacks. According to Table V, both methods obtained similar results for *Amsterdam* dataset. However, for *Santiago* dataset, AD-BoP was less efficient when using Holt-Winters as the forecasting method.

TABLE V
PERCENTAGE OF TRAFFIC WINDOWS LABELED AS ANOMALOUS AT THE FIRST DETECTION OF THE NINE INJECTED ANOMALIES. COMPARISON OF BOTH VERSIONS OF THE AD-BOP METHOD.

	Santiago	Amsterdam
AD-BoP LSTM	5.9%	9.4%
AD-BoP Holt-Winters	58.4%	24.8%

In addition, we studied the performance of both versions of AD-BoP according to their execution time. As mentioned in Section VII-C, when AD-BoP receives a new `.pcap` file, it performs four steps. In the first two steps, AD-BoP extracts the features from the raw file and compares these real features against the values predicted in the previous time interval. These two initial steps are independent of the forecasting method being used, and therefore, the average time needed to decide the presence or absence of an anomaly is the same for both versions of AD-BoP: 7.90 s for *Santiago* dataset and 28.30 s for *Amsterdam* dataset, as stated in Table IV. In contrast, the last two steps are different for each prediction method. In the third step, AD-BoP updates (or refits) the prediction model considering the `.pcap` file just arrived. In the fourth step, the forecasting method performs a one-step ahead prediction of the nine traffic features for the next interval. According to Table VI, Holt-Winters models were faster on average to be updated and perform prediction.

TABLE VI
AVERAGE TIME NEEDED TO UPDATE (OR REFIT) THE PREDICTION MODEL AND PERFORM A ONE-STEP AHEAD PREDICTION.

	Santiago and **Amsterdam**
AD-BoP LSTM	3.8 s
AD-BoP Holt-Winters	2.52 s

Nevertheless, as mentioned in Section VII-C, these two final steps are not needed to label the current traffic window as normal or anomalous. These procedures must be completed before the arrival of the next `.pcap` file, and therefore, their execution time must be bounded by 10 minutes. However, as shown in Table IV, both methods required much less than 10 minutes to complete steps iii and iv. Consequently, the improvement in execution time obtained by Holt-Winters does not have real significance.

These results are an example of a well-known trade-off between prediction goodness and complexity. In our particular case, the reduction in false positives (when using LSTM) is much more relevant than the reduction in the time needed to predict new values (when using Holt-Winters). In addition, as discussed in Sections VII-A, VII-B, and VII-C, AD-BoP (using LSTM) improves the current state-of-the-art for DNS anomaly detection in authoritative TLD name servers. Therefore, using an LSTM model is shown to be appropriate and fit the requirements of the DNS anomaly detection problem.

E. Limitations of the current work

Differently from the two state-of-the-art methodologies used for comparison (CZ.NIC method and QLAD-global), our proposed AD-BoP method requires a first training process to create the LSTM forecast model. Some works point this training methodology as something to be avoided for DNS anomaly detection due to the need for a large amount of historical data to feed the machine learning models [16]. However, we do not consider this a real problem for TLD registry operators, as they can collect all the data directly from the normal operation of their own name servers.

A major concern about this study is that, as we lack labeled data, we could be training our LSTM forecast model using DNS traffic with some anomalies inside. We can then implicitly make the model learn these anomalies and use them to define normal traffic behavior. This is, in fact, not only a consideration for our AD-BoP method but a common concern for general unsupervised methodologies. Because of this, unsupervised network anomaly detection techniques are based on the assumption that only a small percentage of traffic is malicious [29], [30], and that the model learned during training is robust to these few anomalies [31]. Despite this, we claim that learning the normal DNS traffic behavior by using data with some anomalies inside is not necessarily something wrong. This is exemplified in Fig. 11, which shows the number of DNS queries for `A` records received by one of NIC Chile's authoritative name servers during a period of three weeks. In this real example, there is a repetitive anomaly with a clear 24-hour periodicity. As this anomaly presents a highly repetitive pattern, it can actually become part of

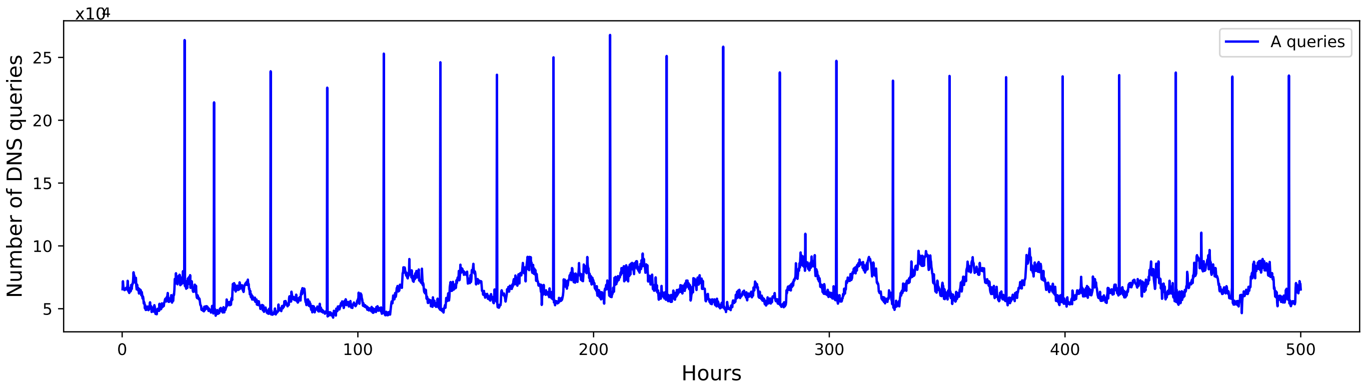


Fig. 11. Number of DNS queries for `A` records received by one of NIC Chile's authoritative name servers during 3 weeks.

the TLD server's expected traffic. In that case, this anomaly will no longer meet the consensual high-level definition of an anomaly, referring to a pattern that does not conform to what is expected [4], [5], [6]. Thus, it is debatable whether the recurrent anomaly eventually became part of normal traffic, considering that a machine learning technique could actually expect its occurrence.

F. Data Availability

The anonymized data used by the DNS anomaly detectors and that support the findings of this study are available on request from the corresponding authors. The data are not publicly available due to privacy concerns inherent to the nature of real DNS traffic.

VIII. Conclusion and Future Work

This paper presented a methodology for Anomaly Detection Based on Prediction (AD-BoP), by using a machine learning model to forecast different portions of the whole DNS traffic. AD-BoP provides a useful and easily explainable methodology to effectively detect DNS anomalies, which could be especially helpful for TLD registry operators to preserve the reliability of their services.

After an exhaustive analysis, the proposed method was demonstrated to improve the state-of-the-art methodologies for DNS anomaly detection in TLD name servers. AD-BoP outperformed these methodologies by efficiently detecting a set of artificially created DNS anomalies, presenting low false positive rates and near real-time execution times.

Differently from other previous works, this study was performed using large amounts of DNS traffic, considering an entire month of normal operation from two authoritative TLD name servers. Thus, this work and its results take on great importance and relevance.

To enhance the readability of our findings, AD-BoP used the same threshold configuration for all the traffic features at detecting anomalies. However, a TLD operator may be interested in assigning different sensitivity configurations to each traffic feature. Therefore, the effectiveness of AD-BoP can be outperformed by selecting different threshold values for each feature.

Acknowledgment

This work was partially funded by the National Agency for Research and Development (ANID) / Scholarship Program / Doctorado Nacional 2019 - 21190450 and by the Millennium Institute for Foundational Research on Data (IMFD).

References

[1] S. Bortzmeyer, "DNS Privacy Considerations," RFC 7626, Aug. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7626.txt

[2] J.-Y. Bisiaux, "Dns threats and mitigation strategies," *Network Security*, vol. 2014, no. 7, pp. 5–9, 2014.

[3] B. Zdrnja, N. Brownlee, and D. Wessels, "Passive monitoring of dns anomalies," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2007, pp. 129–139.

[4] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[5] P. Gogoi, D. Bhattacharyya, B. Borah, and J. K. Kalita, "A survey of outlier detection methods in network anomaly identification," *The Computer Journal*, vol. 54, no. 4, pp. 570–588, 2011.

[6] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[7] Z. Whang and S.-S. Tseng, "Anomaly detection of domain name system (dns) query traffic at top level domain servers," *Scientific Research and Essays*, vol. 6, no. 18, pp. 3858–3872, 2011.

[8] M. Grill, T. Pevnỳ, and M. Rehak, "Reducing false positives of network anomaly detection by local adaptive multivariate smoothing," *Journal of Computer and System Sciences*, vol. 83, no. 1, pp. 43–57, 2017.

[9] S. Yadav, A. K. K. Reddy, A. N. Reddy, and S. Ranjan, "Detecting algorithmically generated domain-flux attacks with dns traffic analysis," *IEEE/Acm Transactions on Networking*, vol. 20, no. 5, pp. 1663–1677, 2012.

[10] R. Villamarín-Salomón and J. C. Brustoloni, "Identifying botnets using anomaly detection techniques applied to dns traffic," in *2008 5th IEEE Consumer Communications and Networking Conference*. IEEE, 2008, pp. 476–481.

[11] Y. Musashi, M. Kumagai, S. Kubota, and K. Sugitani, "Detection of kaminsky dns cache poisoning attack," in *Intelligent Networks and Intelligent Systems (ICINIS), 2011 4th International Conference on*. IEEE, 2011, pp. 121–124.

[12] M. Čermák, P. Čeleda, and J. Vykopal, "Detection of dns traffic anomalies in large networks," in *Meeting of the European Network of Universities and Companies in Information and Communication Engineering*. Springer, 2014, pp. 215–226.

[13] L. Deri, L. L. Trombacchi, M. Martinelli, and D. Vannozzi, "A distributed dns traffic monitoring system," in *2012 8th International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, 2012, pp. 30–35.

[14] J. Qiao. (2017) .nz dns traffic: Trend and anomalies. [Online]. Available: https://blog.nzrs.net.nz/nz-dns-traffic-trend-and-anomalies/

[15] O. Mikle, K. Slanỳ, J. Veselỳ, T. Janoušek, and O. Surỳ, "Detecting hidden anomalies in dns communication," *Sponsoring Institutions*, p. 93, 2011.

[16] P. Robberechts, M. Bosteels, J. Davis, and W. Meert, "Query log analysis: Detecting anomalies in dns traffic at a tld resolver," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018, pp. 55–67.

[17] G. N. V. Sekar, D. Anderson *et al.*, "An empirical evaluation of entropy-based anomaly detection," in *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement,, ACM Press, pp151-156*, 2008.

[18] D. Madariaga, M. Panza, and J. Bustos-Jiménez, "Dns traffic forecasting using deep neural networks," in *International Conference on Machine Learning for Networking*. Springer, 2018, pp. 181–192.

[19] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 240–254, 1994.

[20] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific reports*, vol. 8, no. 1, pp. 1–12, 2018.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] H. Griffioen and C. Doerr, "Taxonomy and adversarial strategies of random subdomain attacks," in *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2019, pp. 1–5.

[23] M. Anagnostopoulos, G. Kambourakis, S. Gritzalis, and D. K. Yau, "Never say never: Authoritative tld nameserver-powered dns amplification," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2018, pp. 1–9.

[24] R. Alonso, R. Monroy, and L. A. Trejo, "Mining ip to domain name interactions to detect dns flood attacks on recursive dns servers," *Sensors*, vol. 16, no. 8, p. 1311, 2016.

[25] NIC Chile. Official registry for the .cl cctld. [Online]. Available: https://www.nic.cl/

[26] NIC Chile. .cl nameservers map. [Online]. Available: https://www.nic.cl/estadisticas/mapaDNS.html

[27] L. Portnoy, E. Eskin, and S. Stolfo, "Intrusion detection with unlabeled data using clustering," in *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001*. Citeseer, 2001.

[28] C. Chatfield, "The holt-winters forecasting procedure," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 27, no. 3, pp. 264–279, 1978.

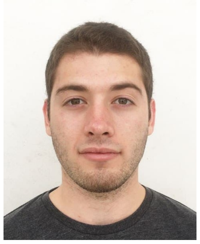
[29] A. A. Ghorbani, W. Lu, and M. Tavallaee, *Network intrusion detection and prevention: concepts and techniques*. Springer Science & Business Media, 2009, vol. 47.

[30] M. Ahmed and A. N. Mahmood, "Novel approach for network traffic pattern analysis using clustering-based collective anomaly detection," *Annals of Data Science*, vol. 2, no. 1, pp. 111–130, 2015.

[31] P. Gogoi, B. Borah, and D. K. Bhattacharyya, "Anomaly detection analysis of intrusion data using supervised & unsupervised approach," *Journal of Convergence Information Technology*, vol. 5, no. 1, pp. 95–110, 2010.

Diego Madariaga received the BSs and MSc degree in computer science from the University of Chile in 2017 and 2019, respectively. He is currently working toward the PhD degree within the framework of a joint thesis supervision between the University of Chile and the Institut Polytechnique de Paris. His current research interests include machine learning for anticipatory networking, user mobility, network privacy/security, and QoS monitoring.

Javier Madariaga is working toward the MSc degree at the Department of Mathematics Engineering of the University of Chile. He is currently working at NIC Chile Research Labs as research assistant. His research interests include optimization and equilibrium, convex optimization theory, machine learning, optimization from big data, and its applications.



Martín Panza is working toward the MSc degree at the University of Chile. He joined NIC Chile Research labs in 2016 where he worked as a research engineer on computer networks projects. He spent a year in the United States on 2019 as a software engineer intern developing vehicular network projects for Cisco Systems. He is currently working on implementing anomaly detection methodologies for DNS traffic, and working for the robotics company Zippedi Inc. as a software engineer.

Javier Bustos-Jiménez received his D.Sc. degree in 2006 from Universite Nice Sophia Antipolis (thesis developed in OASIS team at INRIA). In 2012 he joined NIC Chile Research Labs as Director. His research interests include complex networks, internet protocols, network privacy/security, and data science.

Benjamín Bustos received the doctoral degree in natural sciences from the University of Konstanz, Germany, in 2006. He is an associate professor with the Department of Computer Science, University of Chile. He is head of the PRISMA Research Group, and he is also a researcher with the Millennium Nucleus Center for Semantic Web Research. He leads research projects in the domain of content-based multimedia information retrieval. His research interests include similarity search, 3D object retrieval, multimedia mining, Semantic Web, metric/nonmetric indexing, and pattern recognition.