Detection of illicit software Bot activities over DNS

Bruno Aguiar*, Paulo Salvador*, Pétia Georgieva*
**DETI, University of Aveiro, Portugal*

***Abstract*—DNS is a critical component of the Internet where almost all Internet applications and organizations rely on. Its shutdown can deprive them from being part of the Internet, and hence, DNS is usually the only protocol to be allowed when Internet access is firewalled. The constant exposure of this protocol to external entities force corporations to always be observant of external rogue software that may misuse the DNS to establish covert channels and perform multiple illicit activities, such as command and control and data exfiltration.**
This paper proposes mechanisms to detect malware bots and botnet behaviors on DNS traffic that are robust to encrypted DNS traffic and that ensure the privacy of the involved entities by analyzing the behavioral patterns of DNS communications using descriptive statistics over collected network metrics such as packet rates, packet lengths, and silence and activity periods. After characterizing DNS traffic behaviors, a study of the processed data is conducted, followed by the training of Novelty Detection algorithms with the processed data.
Models are trained with licit data gathered from multiple licit activities, in multiple operating systems, browsers, and configurations. Then, the models were tested with similar data, but containing bot malware traffic. Our tests show that our best performing models achieve detection rates in the order of 99%, and 92% for malware bots using low throughput rates.

***Index Terms*—Machine Learning, DNS, Bot Malware Detection, Botnet Detection, Anomaly Detection, Novelty Detection**

I. Introduction

Malware bots are a major cyber-threat for today's Internet, partially because they can effectively disrupt targeted Infrastructures, but also because malicious actors can simply rent malware services from cybercriminals without much effort, resulting in a large economic damage for corporations and individuals [1] [2]. This rogue software is designed to perform a multitude of illicit activities [3], such as DDoS attacks, data exfiltration, malware dissemination, bitcoin mining, identity theft, and so forth, while taking control over the device's operations. The term "Bot" derives from the word "Robot", which refers to a physical or a virtual agent capable of following a set of actions, or instructions, in an automated way, mimicking, or replacing, the normal behavior of a human.

Being a bot by itself is not malicious. Because of their automated nature, bots can be used for licit and helpful purposes, such as performing customer services or indexing content for search engines, operating much faster than humans. However, like many other software applications, bots can also be used to perform malicious purposes. The Malicious bot may not be fully automated, but have an external entity controlling it. This entity is called the botmaster, and a Command and Control (C&C) channel can be established with his bot, allowing the botmaster to send instructions (commands) and receive data from the infected machine, and being an important communication infrastructure to coordinate, manage or suspend attacks [4].

The main goal of this paper is to provide a robust and data privacy focused framework that detects if a device has been infected with a malware bot under a corporate network. The large majority of current solutions are invasive and often deal with Deep Packet Inspection (DPI) techniques that might contain actual data, for instance, the examination of DNS payloads, such as using information entropy on DNS queries and the identification of TCP and UDP packet headers. Naturally, this collection of personally identifiable data rise some privacy concerns. Furthermore, the usage of the DNS protocol by malware bots has been largely overlooked in the literature, specially when the topic is low throughput malware communications and the advent of licit and illicit DNS traffic over encrypted protocols.

Our solution only uses the behavioral and temporal aspects of licit and illicit usages of DNS aiming to be robust to encrypted DNS traffic, and in the contemplation of data privacy laws, our solution only uses IP addresses to distinguish between the device being monitored and the DNS server, important for the collection of upstream and downstream metrics.

A consistent pipeline will be presented throughout this paper, showcasing how licit and illicit data were captured and processed into meaningful features, which are used by the Anomaly Detection algorithms. Since collecting all kinds of illicit activities is impractical, we created some of the possible activities and behaviors that malware bots may have, and we used Novelty Detection models to identify novelties and unseen phenomena in data, and therefore, these classifiers only train with anomaly-free samples.

The remainder of this paper is organized as follows. Section II addresses the problem regarding malware bots over DNS, relevant to the present context. Section III addresses the Anomaly Detection background. Section IV showcases the proposed framework, followed by the testing scenarios in section V, and results in section VI. Finally, we address our conclusions and future work in section VII

II. The problem

Malware bots that use DNS to perform their illicit activities usually use DNS Tunneling tools to embed their exfiltrated data and commands in DNS payload fields. Data can be encoded in the host name, and the malware bot can encode up to 63 characters in each label, where each character must be a letter, a number or a hyphen, and the characters must be all upper case or lower case. To effectively send data over DNS and to comply with these restrictions, encoding schemes such as Base32 and Base64 are often used. DNS queries are sent to a rogue DNS server that forwards the received data to the botmaster. This forwarding is exemplified in Figure 1, that depicts an exfiltration of a password.

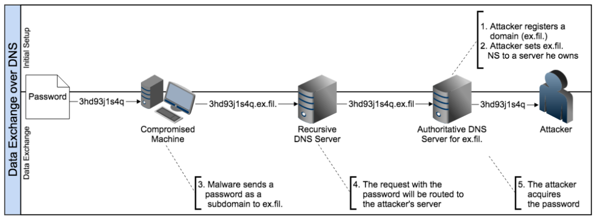
Data Exchange over DNS
Initial Setup
Data Exchange
1. Attacker registers a domain (ex.fil.)
2. Attacker sets ex.fil. NS to a server he owns
Password
3hd93j1s4q
3hd93j1s4q.ex.fil.
3hd93j1s4q.ex.fil
3hd93j1s4q
Compromised Machine
Recursive DNS Server
Authoritative DNS Server for ex.fil.
Attacker
3. Malware sends a password as a subdomain to ex.fil.
4. The request with the password will be routed to the attacker's server
5. The attacker acquires the password

Fig. 1: Misuse of DNS for data exfiltration purposes. Source: Detection of malicious and low throughput data exfiltration over the DNS protocol [5].

A. The DNS threat landscape

Asaf Nadler *et al* [5] defined two possible scenarios for the use of DNS as a covert channel: as low throughput data exfiltration and high throughput DNS tunneling. Each scenario has different communication patterns. DNS Tunneling software is typically more reliable, as it relies in frequent *keep-alive* messages to inform that the client software is alive and waiting for commands. This software is used to establish bi-directional and interactive channels, and they usually produce *lengthy* messages, as packets with larger sizes decrease the likelihood of network congestion.

In low throughput exfiltration, malware exchange queries less frequently, in a more *opportunistic* and *unexpected* manner, and their packet sizes are better crafted as well, meaning greater evasion from traditional security measures. Their connection is typically unidirectional, receiving from the remote server, at most, acknowledge messages. These slow and stealthy attacks are planned to be performed over longs periods of time, like weeks or months, and are not usually detected via traditional defense systems.

B. Major Command and Control approaches

There are two major approaches to deliver commands to malware bots: by either using *pull* mechanisms or *push* mechanisms [6], [7]. In *pull* mechanisms, bots retrieve commands actively from a C&C server, where botmasters publish them, following a publish–subscribe pattern, while in *push* mechanisms bots establish a session with C&C servers and passively wait for the arrival of commands. The former is typically used in centralized topologies, especially in HTTP botnets, while the latter is commonly used in P2P bots that relay commands and in IRC-based bots. In DNS botnets the study of these two types of C&C communications have been overlooked, thus one should beware of both implementations.

III. Background

This section focus on showcasing the techniques used for detecting malware bot activities, namely unsupervised dimensionality reduction methods and Novelty models for Anomaly Detection.

A. Unsupervised Dimensionality Reduction

Unsupervised Dimensionality Reduction is a process of transforming high dimensional data in low dimensional representations without the help of the target value, aiming to address the *Curse of Dimensionality* problem while maintaining as much of the variance as possible [8]. Compressing the number of dimensions in a dataset has several benefits, as it can lead to lesser training times and memory usage, faster model convergence, better data visualization, and even greater model performance. Unlike Feature Selection methods, Unsupervised Dimensionality Reduction techniques do not make Anomaly Detection models to be biased torwards a particular set of anomalies.

The most popular Unsupervised Dimensionality Reduction technique is the Principal Component Analysis (PCA) and it works by finding the basis vectors of the orthonormal subspace where data vary the most [9], however other unsupervised approaches can also be used, such as removing the pair of features that are highly correlated.

B. Novelty Detection

Novelty detection models model a representation of normality during the training phase and check for anomalies, or novelties, in the test set, and some of them can be classified into density-based models and boundary-based models:

1) Density-based models: Density-based models assume that the training data is generated by some underlying probability distribution, of which can be estimated during the training phase, evaluating the log-likelihood of a test sample in conformity with that model. Density estimation techniques can be divided into two major groups: the *parametric* methods, where they fit a given model on data assuming that data come from a population that can be modelled by a probability distribution with a finite set of parameters, and *non-parametric* methods, where they do not make assumptions about the underlying probability distribution, and so, they can adapt very easily to the complexity of data that do not conform to a particular known distribution [10]. In parametric methods, the Gaussian distribution is often used for Anomaly Detection, and for datasets whose distributions are not convex and unimodal, a mixture of Gaussians are commonly used, also called as Gaussian Mixture Models (GMM), while in non-parametric methods, Kernel Density Estimation (KDE) is the preferred model.

2) Boundary-based models: Boundary-based methods for Anomaly Detection optimize a closed boundary around the training set in an effort to construct a representation of normality might be too demanding, specially when the data to do so is scarce and only the data boundaries are enough. The volume of the boundary is not always minimized, but most algorithms have a strong bias towards minimal volumes. Although Boundary-based methods require less samples than Density-based methods, they rely heavily on the distances between objects, therefore, data scaling plays an important role in the viability of these methods. Common Boundary-based algorithms are the Local Outlier Factor (LOF), that estimates the local density of test points to data points that are its neighbors and, and the One-Class SVM, that learns to minimize the volume of the data-enclosing hyperplane of the single class of instances in training data, considering all the other samples outside that hyperplane as outliers.

IV. Framework for bot malware detection

This section presents the proposed framework the detection of malware bots. Contrary to other solutions for bot malware and botnet detection, which resorts in Deep Packet Inspection techniques, such as analyzing the payload fields of DNS queries, we are proposing a privacy-focused solution for the detection of malware bots and illicit activities over DNS in general that are also robust to encrypted data.

In order to have a proper overview of the proposed framework, the overall system is presented as a series of steps to be taken for bot malware detection.

A. Data collection

Using open datasets with licit and malware communication traces can be challenging. Public DNS datasets with C&C and data exfiltration traces can be limited in variety, the reliability of the *ground-truth* is hard to be ensured, since licit and illicit DNS traffic can be encrypted, and new normal and malware bot activity patterns are continually emerging as Internet technology evolve. Labeling samples with mixed traffic is also a complex and time-consuming task, since one often needs to combine DPI techniques and expert knowledge for a successful labeling, and yet, they cannot ensure that network attacks are fully identified [11].

A better approach for a perfect *ground-truth* is to emulate malware traffic in a Laboratory environment, providing reliable data to Machine Learning models, where their predictive capacity could, otherwise, be degraded upon their deployment in a real world scenario, if malware bot attacks and normal behaviors were mislabeled. This emulation must gather a vast variety of possible illicit communication behaviors over multiple malware bot activities. Furthermore, the licit traffic must also cover a wide range of possible normal behaviors, and it must be produced under a controlled environment to ensure that the terminal device is not infected with some sort of virus, worm, Trojan horse or an active botnet.

1) Data collection under data protection regulations: With the advent of data protection regulations, cyber-security professionals must use as little as Personal and Identifiable Information (PII) as possible. In our case, since we are only interested in identifying the misuse of the DNS to establish covert channels, or to exfiltrate data, by solely monitoring the DNS communications of individual terminal devices, and since IP addresses are classified as PII, as they can reveal personal data such as geographical localization of individuals, [12] we only use IP addresses to solely differentiate if a given processed IP is from a set of known, and permitted, DNS servers or is an end device, which is turn, is useful for the collection of upstream and downstream network metrics, such as, the upstream and downstream metrics of packet rates and average packet lengths in each sampling interval.

2) Emulating network behaviors of malware bots: The emulation of illicit DNS usages passes through using DNS Tunneling tools that embed data in DNS queries and deliver DNS requests and responses between the client and the C&C server. The C&C server can afterwards forward the received data to the botmaster. The three most common DNS Tunneling tools are Iodine, which allows Base32, Base64, Base64u and Base128 encoding schemes, and several DNS RRs are supported, and DNS2TCP, which offers greater throughput rates than Iodine, but limited configurations.

B. Feature extraction with observation windows

The collected network traffic metrics may be numerical, but they have no profound insights, and do not show any meaningful attributes or relationships whatsoever [13]. Notwithstanding, network traffic metrics can be properly processed and used to extract patterns with real, underlying meaning. For instance, extracting descriptive statistics, such as measures of central tendency and measures of variability of network metrics over a time period, from licit and illicit DNS traffic might reveal divergent distributions that could further indicate that a given machine is infected and being part of a botnet.

Observations windows is a technique that allows one to derive relevant features from a stream of metrics, or events, over consecutive time intervals [14], i.e., windows. The computed features describe events that occurred in a given timestamp, representing an observation, and multiple observations form a characterization of an entity, also called as profile, enabling one to group entities that show a similar behavior and to detect illicit entities that reveal behavioral deviations from the known group. A common window technique for a robust and rich to feature extraction is to use sliding windows with multiple observation sizes, creating different meaningful representations and perceptions about the collected data by varying the observation time interval. The dataset must be split according to the larger window size, as well as the specified sliding value, and for every sub-window at sample i the relevant features are computed. As showcased in Figure 2, for malware bot detection, the mean, 5% trimmed mean, standard deviation, 85, 90, 95 and 99 quantile, and maximum values with regard to the collected traffic metrics and the activity and silence periods are computed. For the latter, also the number of silence and activity periods are retained. These statistics captured from different sub-windows are then joined into a single feature vector, forming the dataset.

C. Data scaling

Because our data was produced in controlled environments, they should be outlier and noise-free, however, as features came from multiple metrics, they are in different ranges. Most predictive models do not generalize well on features whose scales differ vastly, therefore their ranges should be either scaled or transformed so that each feature contributes equally to the final predictive result.

The two most common feature scaling methods are Normalization and Standardization. In Normalization, data are transformed into a homogeneous range of values, often between 0

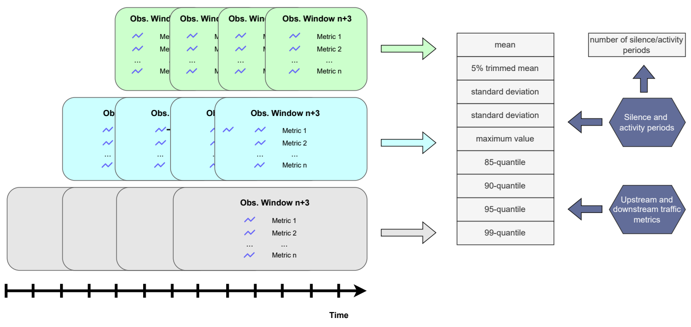

Fig. 2: Diagram of features extracted from multiple sub-windows.

and 1, or -1 and 1, it is typically used when the underlying distribution unknown, or when the data do not follow a bell-shaped curve, while in Feature Standardization, the values of each feature are set with a zero-mean and a standard deviation equals to one.

After choosing the scaling technique, the mean and standard deviation of features, with regard to Standarization, or the maximum and minimum values of features, with regard to Normalization, should be computed. These computations **must be done only with the training set**, while the transformation is applied to both the training and test sets.

D. Dimensionality Reduction

Although data scaling guarantees that features have an equal contribution, it does not imply that these features are equally important for the decision process of the model. Most likely, some extracted features may be irrelevant, and/or redundant, to the class value. Furthermore, having many features but a low number of samples may decrease the performance of Machine Learning models, therefore unsupervised dimensionality reduction techniques, such as the PCA, may be considered. The application of dimensionality reduction techniques in the data must be done in the same way as the aplication of data scaling techniques.

E. The Learning Process and model evaluation

Once the data is preprocessed, it is finally ready to be used by Anomaly Detection models. First, we need to split the dataset with licit instances into training and test sets. The main reason to have a test dataset is to test the model against new and unseen malware bot traffic, therefore the test set must contain, not only licit samples, but also samples from new and unseen bot malware attacks.

When evaluating different hyper-parameter settings for models using a single test set, there is risk of overfitting on the test set, as parameters can be tweaked until the model fits optimally on that particular set, not generalizing well on a real-world scenario. Therefore, a better practice for classic Machine Learning techniques is to use Cross-Validation techniques, such as the k-fold Cross-Validation with validation and test sets, that use different partitions of the data, in a iterative way, to train the model and choose the best hyper-parameter settings for more accurate out-of-sample and overfitting estimations, specially when data is scarce, or the number of features is large.

The validation and test sets should have an equal amount of licit and malware bot samples, in order to not get skewed performance metrics. An example of a splitting process at fold k is illustrated in Figure 3.

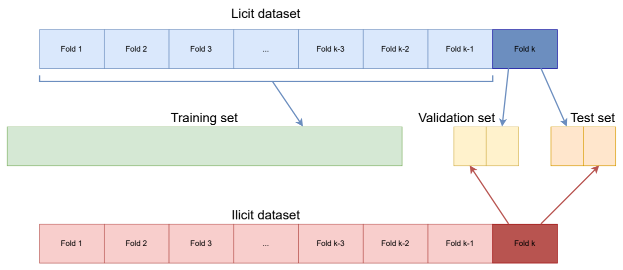

Fig. 3: Illustration for the k-fold Cross-Validation with validation and test set at iteration k

At each fold, the k-fold Cross-Validation with validation and test set approach selects the hyper-parameter setting that obtained the best weighted F1-Score, and an additional evaluation is made with the test set, obtaining the weighted F1-Score and the False Positive Rate (FPR). To choose between different estimators, the results of weighted F1-Scores and FPRs must be averaged and a 95% confidence interval must be computed.

V. Experimental Setup

This section presents the experimental setup and scenarios , with the objective to detect illicit usages of DNS.

A. Licit traffic collection

The licit DNS traffic was collected using Wireshark, while performing multiple activities in a web browser, such as reading the news, studying, using social networks and consuming media, and streaming music via Spotify's own application, for, around, 3 hours straight. These use cases were produced under the three major operating systems: Linux, macOS and Windows. In some Linux Ubuntu traces the option to send diagnostic data is activated, while in others the machine do not send any diagnostic data whatsoever. In Windows and macOS systems this option is required. For each operating system, the browser used in captures was the pre-installed one, namely Firefox in Linux Ubuntu, Safari in macOS and Microsoft Edge in Windows. In order to cover a wide variety of licit use cases, some of them use ad blockers, while others don't, and in one use case, even the JavaScript was disabled and media larger than 50kb was blocked. To prevent false alarms from encrypted, but licit traffic, two use cases in Linux Ubuntu using DNS over TLS (DoT), one with the ad blocker active and other without using ad blockers, were also captured.

B. Emulation of malware bot attacks

For the emulation of malware bot attacks, we partially simulated a corporate network using GNS3 with an infected computer, a DNS Server and a Router, a public DNS server used by the botmaster to register a domain for the further establishment of the covert channel, a C&C server and the botmaster's computer. The DNS servers were implemented with BIND9, using the default settings, and static routing was used at the core of the simulation. All devices are accessible via SSH using a NAT interface, and they can also be connected to real world Internet using a TAP interface. The simulated environment is depicted in Figure 4. As a side note, since we did not configure any mapping methods between private and public corporate addresses, all nodes in the Internet simulation know the company's private addresses, although, this is not a big issue as we are only interested in monitoring the link between the infected machine and its corporate DNS server.

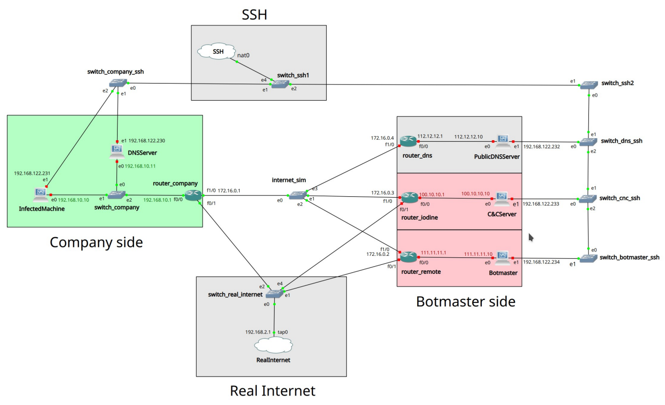

Fig. 4: Simulated corporate environment for the emulation of botnet attacks

To exfiltrate data and send commands over DNS, the two most common DNS tunneling tools to emulate botnet traffic were chosen: Iodine and DNS2TCP. In Iodine, we decided to use the `NULL` RR type as it provides the faster upstream throughput among the available RRs, and we configured the `keep-alive` messages to have a frequency lower enough to not get `SERVFAIL` messages, i.e. 9 seconds, while in DNS2TCP the RR used is the `TXT` query type with Base64.

To have a broad diversity of possible attacks, 6 scenarios were a botnet can try to emulate a normal behavior, or to blend with normal traffic without being detected, were produced: the Scenario 1, were we emulate a C&C channel with the Push mechanism over DNS using the original Iodine and DNS2TCP tools and , the Scenario 2, were we emulate exfiltration of data with different probability distributions that resemble licit behaviors using the original Iodine and DNS2TCP tools, two more advanced scenarios, Scenario 3 and Scenario 4, were we emulate a C&C channel with the Push mechanism and exiltration of data with a modified version of the DNS2TCP tool to control the packet sizes and using empirical non-parametric distributions taken from the collected licit traffic to generate realistic packet rates and lengths, the Scenario 5, were we emulate stealth C&C traffic with Pull mechanisms, and finally the Scenario 6, were we emulate a slow and low exfiltration of data using the modified version of the DNS2TCP. As Scenarios 5 and 6 generate little data over time, they were both mixed with licit traffic.

C. Dataset exploration

The datasets for each scenario were produced with a sliding window of 70 minutes with 2 seconds of sliding distance and sub-windows with 450, 750, 2250 and 3375 seconds, and have around 30 hours worth of data, each one. For testing purposes, the Scenario 1 and 2 were grouped in *anomalies that use standard behaviors*, Scenarios 3 and 4 were grouped in *anomalies that mimic licit traffic* and Scenarios 5 and 6 were grouped in *anomalies mixed with licit traffic*. *Keep-alive* messages are only present in the dataset that use standard behaviors. As an example, if we compare illicit traffic emulated with different behavioral approaches (Figures 5, 6, and 7), such as, malware bots that use standard behaviors, malware bots that mimic licit traffic and malware bots that mix with licit traffic, we will see that using standard behaviors, i.e. using parametric models, often produce traffic with more divergent distributions and that low throughput malware traffic when mixed with licit traffic can be very effective in deluding the current security systems.

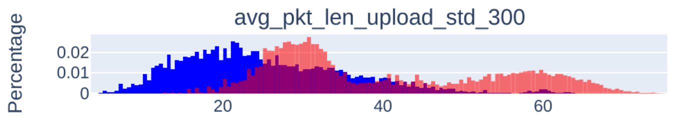

Fig. 5: A comparison between licit (in blue) and illicit (in red) data points without *keep-alive* messages that was produced with standard behaviors

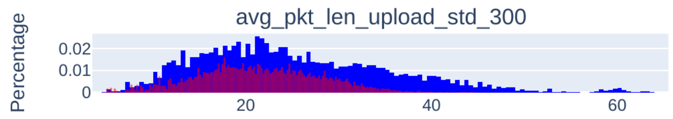

Fig. 6: A comparison between licit (in blue) and illicit (in red) data points without *keep-alive messages* that was produced mimicking the licit traffic

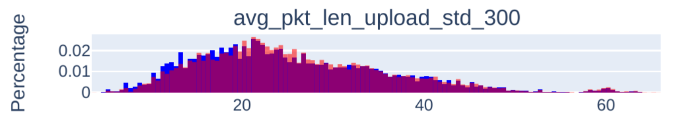

Fig. 7: A comparison between licit (in blue) and illicit (in red) data points without *keep-alive messages* that was produced mixed with licit traffic

VI. Classification Results

The final step is to train each AD model with all features, and the selected unsupervised dimensionality reduction techniques. Beforehand, we used the standardization as feature scaling for our data. The tested models were GMM, KDE, LOF, and One-Class SVM, using the Scikit-Learn Python implementations, running on a Jupyter Notebook and they were trained without any dimensionality reduction method and with PCA with a variance threshold of 0.85.

For each dataset and each model, we took the data points from a from a random illicit capture and merged them with an equal amount of licit samples drew from the training data. This set was used to test the best model found in k-fold Cross-Validation with new and unseen bot malware traffic, simulating, therefore, a real-world scenario. The rest of the licit and illicit data points were used as training sets for the k-fold Cross-Validation, using 10 folds ($k = 10$), although illicit instances were only used to create the validation sets for hyper-parameter optimization, and the test sets to evaluate the best model in each fold, both with also an equal amount of licit samples.

A. Classification results with keep-alive messages

Table I shows that most of the tested AD models had some difficulties in correctly classifying the test data, with GMM having the least optimal detection rate. In contrast, One-Class SVM had excellent performance, having the best F1-Score on the 10-fold Cross-Validation and no false positives whatsoever.

		WITHOUT Dimensionality Reduction			
Type of anomaly		**GMM**	**KDE**	**LOF**	**One-Class SVM**
standard behaviors	*F1-Score*	75.80 ± 0.82	76.66 ± 0.48	78.08 ± 1.42	$\mathbf{98.87 \pm 0.21}$
	FPR	31.41 ± 0.65	30.74 ± 0.40	29.46 ± 1.28	$\mathbf{0.0 \pm 0.0}$

TABLE I: Average results on 10-fold test sets without Dimensionality Reduction for datasets with *keep-alive* messages and with a sliding window of 70 minutes (in percentage, with a 95% confidence interval)

Interestingly, models that performed the worst in Cross-Validation, actually were the ones that had the best F1-Score on the test set with new and unseen bot malware traffic, as showcased in Table II. Nevertheless, One-Class SVM also a showed consistent F1-Score on this test, and all models didn't have any false positives.

		WITHOUT Dimensionality Reduction			
Type of anomaly		**GMM**	**KDE**	**LOF**	**One-Class SVM**
standard behaviors	*F1-Score*	**100.0**	**100.0**	99.97	98.77
	FPR	**0.0**	**0.0**	0.0	0.0

TABLE II: Results of tests on new and unseen bot malware traffic without Dimensionality Reduction for datasets with *keep-alive* messages and with a sliding window of 70 minutes

When using PCA (Tables III and IV), the F1-Score for all models increased, while the FPR decreased. One-Class SVM was once again the model with the highest F1-Score and the lowest FPR on the 10-fold Coss-Validation, and the model that had more consistency between Cross-Validation test sets and the test set with new and unseen bot malware traffic.

		WITH PCA			
Type of anomaly		**GMM**	**KDE**	**LOF**	**One-Class SVM**
standard behaviors	*F1-Score*	78.39 ± 1.28	79.73 ± 0.43	81.30 ± 0.87	$\mathbf{99.34 \pm 0.14}$
	FPR	29.20 ± 1.09	28.05 ± 0.40	26.51 ± 0.84	$\mathbf{0.0 \pm 0.0}$

TABLE III: Average results on 10-fold test sets with PCA for datasets with *keep-alive* messages and with a sliding window of 70 minutes (in percentage, with a 95% confidence interval)

		WITH PCA			
Type of anomaly		**GMM**	**KDE**	**LOF**	**One-Class SVM**
standard behaviors	*F1-Score*	**100.0**	**100.0**	**100.0**	99.16
	FPR	**0.0**	**0.0**	**0.0**	0.0

TABLE IV: Results of tests on new and unseen bot malware traffic with PCA for datasets with *keep-alive* messages and with a sliding window of 70 minutes

B. Classification results without keep-alive messages

Results shown in Table V state that most of the tested AD models perform well on the majority types of anomalies. For illicit traffic that uses standard behaviors, or that mimicks licit traffic, KDE is the model that shows the best F1-Score and a very low FPR, nonetheless One-Class SVM, or One-Class SVM, also shows an ideal FPR.

In anomalies mixed with licit traffic and when all types of anomalies are present, however, AD models had a decrease in performance, with worst F1-Scores and high FPRs. Nonetheless, KDE shows the best F1-Score and FPR in anomalies mixed with licit traffic, and when all anomalies are present, LOF shows the best F1-Score and One-Class SVM the best FPR. One common aspect to the majority of the results is that models with lower F1-Scores often show higher FPRs, which means that they often produce false alarms, even though they are able to identify most of the anomalies (high Recall).

		WITHOUT Dimensionality Reduction			
Type of anomaly		**GMM**	**KDE**	**LOF**	**One-Class SVM**
standard behaviors	*F1-Score*	97.83 ± 1.27	$\mathbf{99.99 \pm 0.01}$	99.75 ± 0.09	98.86 ± 0.21
	FPR	3.95 ± 2.29	$\mathbf{0.0 \pm 0.0}$	0.10 ± 0.08	$\mathbf{0.0 \pm 0.0}$
mimicking licit traffic	*F1-Score*	99.08 ± 0.71	$\mathbf{99.93 \pm 0.05}$	98.95 ± 0.23	97.14 ± 0.32
	FPR	1.29 ± 1.28	$\mathbf{0.06 \pm 0.07}$	0.82 ± 0.41	3.46 ± 0.44
mixed with licit traffic	*F1-Score*	88.93 ± 1.56	$\mathbf{92.28 \pm 0.41}$	81.91 ± 0.72	84.26 ± 0.73
	FPR	14.41 ± 2.88	$\mathbf{4.97 \pm 0.80}$	17.36 ± 1.78	12.93 ± 0.88
all anomalies	*F1-Score*	75.10 ± 1.49	88.03 ± 0.85	$\mathbf{89.44 \pm 0.55}$	87.69 ± 0.52
	FPR	31.89 ± 1.17	18.04 ± 1.36	11.77 ± 1.52	$\mathbf{6.58 \pm 2.38}$

TABLE V: Average results on 10-fold test sets without Dimensionality Reduction for datasets without *keep-alive* messages and with a sliding window of 70 minutes (in percentage, with a 95% confidence interval)

The results of tests on new and unseen bot malware traffic (Table VI) show that, once again, KDE was the model with the best F1-Scores and FPRs for anomalies with stantard behaviors and anomalies that mimic licit traffic. When the anomalies were mixed with licit traffic, GMM was the model with the best F1-Score and the only without any false alarms. When all anomalies are present, GMM was also the model with the best F1-Score, but all models were able to correctly classify every licit data point.

When PCA is active (Table VII), keeping 85% of variance, some models slightly increased their performance, while others slightly decreased. In anomalies that use standard behaviors, that mimic licit traffic, and that are mixed with licit traffic, KDE continues to be the model that has the best F1-Score, however, its FPR slightly increased. The FPR of One-Class SVM remains to be ideal for anomalies with standard

Type of anomaly		GMM	KDE	LOF	One-Class SVM
standard behaviors	*F1-Score*	99.95	**100.0**	99.20	98.97
	FPR	**0.0**	**0.0**	1.28	**0.0**
mimicking licit traffic	*F1-Score*	99.84	**99.98**	99.19	89.04
	FPR	**0.0**	**0.0**	**0.0**	16.82
mixed with licit traffic	*F1-Score*	**95.96**	89.48	78.87	79.52
	FPR	**0.0**	8.69	22.03	20.30
all anomalies	*F1-Score*	**99.85**	98.07	94.28	89.29
	FPR	**0.0**	**0.0**	**0.0**	**0.0**

TABLE VI: Results of tests on new and unseen bot malware traffic without Dimensionality Reduction for datasets without *keep-alive* messages and with a sliding window of 70 minutes

behaviors. When we tested the chosen AD models with all anomalies, One-Class SVM obtained the best F1-Score. Its FPR was also a bit high, although lesser than the best FPR when bot malware traffic is mixed with licit traffic.

		WITH PCA			
Type of anomaly		GMM	KDE	LOF	One-Class SVM
standard behaviors	*F1-Score*	99.42 ± 0.46	$\mathbf{99.77 \pm 0.17}$	98.41 ± 0.23	98.26 ± 0.16
	FPR	0.94 ± 0.90	0.36 ± 0.33	1.50 ± 0.3	$\mathbf{0.0 \pm 0.0}$
mimicking licit traffic	*F1-Score*	98.07 ± 0.31	$\mathbf{99.43 \pm 0.09}$	96.71 ± 0.39	95.24 ± 0.38
	FPR	2.24 ± 0.60	$\mathbf{0.35 \pm 0.15}$	2.27 ± 0.66	3.30 ± 0.31
mixed with licit traffic	*F1-Score*	70.67 ± 0.74	$\mathbf{92.25 \pm 0.54}$	82.58 ± 0.73	85.66 ± 0.58
	FPR	29.28 ± 1.77	$\mathbf{5.86 \pm 0.94}$	17.75 ± 2.11	12.26 ± 0.59
all anomalies	*F1-Score*	86.72 ± 0.81	82.16 ± 0.94	88.53 ± 0.48	$\mathbf{88.65 \pm 0.47}$
	FPR	18.57 ± 1.39	25.49 ± 1.02	11.71 ± 1.41	$\mathbf{4.19 \pm 0.60}$

TABLE VII: Average results on 10-fold test sets with PCA for datasets without *keep-alive* messages and with a sliding window of 70 minutes (in percentage, with a 95% confidence interval)

When the above models were tested against new and unseen bot malware traffic (Table VIII), results where in pair with the Cross-Validation scores, with FPRs being ideal most of the time, however, in anomalies mixed with licit traffic, FPRs where higher than those reported in Cross-Validation. KDE was the model that performed best in all types of anomalies.

Type of anomaly		GMM	KDE	LOF	One-Class SVM
standard behaviors	*F1-Score*	99.68	**99.76**	99.26	97.98
	FPR	**0.0**	**0.0**	**0.0**	**0.0**
mimicking licit traffic	*F1-Score*	99.17	**99.62**	98.24	96.69
	FPR	**0.0**	**0.0**	**0.0**	**0.0**
mixed with licit traffic	*F1-Score*	73.68	**90.35**	71.09	82.87
	FPR	27.50	**10.01**	32.36	16.61
all anomalies	*F1-Score*	97.39	**99.02**	93.49	89.49
	FPR	**0.0**	**0.0**	**0.0**	**0.0**

TABLE VIII: Results of tests on new and unseen bot malware traffic with PCA for datasets without *keep-alive* messages and with a sliding window of 70 minutes

VII. Conclusion And Future Work

This paper proposed a framework for Network-based malware bot detection by monitoring the terminal devices and using Novelty Detection algorithms to evaluate the accuracy of the proposed solution in detecting illicit usages of DNS. The obtained results showed that by using Novelty Detection, it is possible to detect the majority of illicit usages with high accuracy and very low false positives, however, for malware bots that generate traffic with very low throughput rates, although the F1-Scores of the best models were satisfactory, further research in reducing the number of false positives is needed. As future work, it would be interesting to test this low throughtput malware traffic with bigger window sizes for a very low FPR, or with Deep Anomaly Detection models, such as Neural Generative Models and Normalizing Flows, where they might get satisfactory FPRs with smaller observation windows.

It would also be interesting to have proper DNS Tunneling tools specifically designed to emulate licit DNS usages with realistic packet sizes and higher configuration capabilities, such as the possibility to configure RRs, encoding schemes, and disable keep-alive messages.

References

[1] C. Wilson, “Botnets, cybercrime, and cyberterrorism: Vulnerabilities and policy issues for congress.” LIBRARY OF CONGRESS WASHINGTON DC CONGRESSIONAL RESEARCH SERVICE, 2008.

[2] K. Alieyan, A. ALmomani, A. Manasrah, and M. M. Kadhum, “A survey of botnet detection based on dns,” *Neural Computing and Applications*, vol. 28, no. 7, pp. 1541–1558, 2017.

[3] M. Feily, A. Shahrestani, and S. Ramadass, “A survey of botnet and botnet detection,” in *2009 Third International Conference on Emerging Security Information, Systems and Technologies*, 2009, pp. 268–273.

[4] A. Karim, R. B. Salleh, M. Shiraz, S. A. A. Shah, I. Awan, and N. B. Anuar, “Botnet detection techniques: review, future trends, and issues,” *Journal of Zhejiang University SCIENCE C*, vol. 15, no. 11, pp. 943–983, Nov 2014. [Online]. Available: https://doi.org/10.1631/jzus.C1300242

[5] A. Nadler, A. Aminov, and A. Shabtai, “Detection of malicious and low throughput data exfiltration over the dns protocol,” *Computers & Security*, vol. 80, pp. 36–53, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404818304000

[6] P. Wang, L. Wu, B. Aslam, and C. C. Zou, “A systematic study on peer-to-peer botnets,” in *2009 Proceedings of 18th International Conference on Computer Communications and Networks*. IEEE, 2009, pp. 1–8.

[7] B. Choi, S.-K. Choi, and K. Cho, “Detection of mobile botnet using vpn,” in *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. IEEE, 2013, pp. 142–148.

[8] R. Zebari, A. Abdulazeez, D. Zeebaree, D. Zebari, and J. Saeed, “A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction,” *Journal of Applied Science and Technology Trends*, vol. 1, no. 2, pp. 56–70, 2020.

[9] H. Abdi and L. J. Williams, “Principal component analysis,” *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[10] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, “A review of novelty detection,” *Signal processing*, vol. 99, pp. 215–249, 2014.

[11] M. R. Oliveira, J. Neves, R. Valadas, and P. Salvador, “Do we need a perfect ground-truth for benchmarking internet traffic classifiers?” in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 2452–2460.

[12] L. Böck, M. Fejrskov, K. Demetzou, S. Karuppayah, M. Mühlhäuser, and E. Vasilomanolakis, “Processing of botnet tracking data under the gdpr,” *Computer Law & Security Review*, vol. 45, p. 105652, 2022.

[13] E. Adi, A. Anwar, Z. Baig, and S. Zeadally, “Machine learning and data analytics for the iot,” *Neural Computing and Applications*, vol. 32, no. 20, pp. 16 205–16 233, 2020.

[14] Azelcast. (2022) Sliding window aggregation. [Online]. Available: https://docs.hazelcast.com/hazelcast/5.1/architecture/sliding-window