Robust and Efficient DNS Data Exfiltration Detection

Deepank Kartikey[a], Romario Vaz[b], Saket Dawgotra[c], Siddhant Tiwari[d]

[a] University of Ottawa, Ottawa, Canada, dkart060@uottawa.ca

[b] University of Ottawa, Ottawa, Canada, rvaz082@uottawa.ca

[c] University of Ottawa, Ottawa, Canada, sdawg084@uottawa.ca

[d] University of Ottawa, Ottawa, Canada, stiwa017@uottawa.ca

Abstract

The Domain Name System (DNS) protocol is often exploited to leak confidential information from enterprises. As it is an essential part of the internet, DNS traffic is typically not monitored by firewalls, and can act as a channel for malicious actors to encode sensitive information and exfiltrate this data to a server they control. In this paper, we build upon an existing algorithm (Mahdavifar et al., 2021) that attempts to detect data exfiltration over DNS as quickly and accurately as possible. We attempt to analyze and improve the accuracy, speed, and robustness of the approach. To do this, we first analyze the properties of DNS exfiltration queries and identify characteristics of queries that are more likely to be benign or malicious, and subsequently use these insights for feature engineering. Then, we use statistical feature reduction techniques to reduce the time taken for detection, while maintaining accuracy. We also explore the use of machine learning model ensembles to improve upon the performance of the approach. As there are many ways to encode data to exfiltrate sensitive information, we evaluate the robustness of our proposed approach when dealing with other kinds of DNS data exfiltration, using data generated from various other DNS exfiltration tools and queries generated from malware.

Keywords

Cybersecurity, Machine Learning, DNS Exfiltration, Malware, Feature Engineering, Artificial Intelligence

1. Introduction

DNS (Domain Naming System) allows users to connect to the internet by translating human readable domain names to machine readable IP addresses. DNS is a critical part of the internet and is designated the “phonebook of the

internet." DNS exfiltration attacks are a type of malicious attack where data from a compromised DNS server is sent to an external server.

The general process of DNS exfiltration occurs is that a cybercriminal gains access to a network and installs malware to control the network and access data. The malware is configured to send data via DNS requests to a remote server. These DNS requests are typically disguised as legitimate traffic through some encoding algorithms like Base64, so they can pass through network security controls unnoticed. This DNS subdomain does not exist in any server and so it is directly forwarded to its designated server and the adversary resolves the IP address. This enables the attacker to decode the query and choose a response query which appears benign to the victim machine. Since DNS is such a fundamental part of the Internet system and since it handles a huge volume of queries, data exfiltration through DNS is typically allowed to pass through firewalls and other security controls, exfiltration can occur without being detected. The data is collected by the remote server, through which the cybercriminal can extract the exfiltrated data.

In this project we build upon the work by Mahdavifar et al. (2021), by attempting to improve three main aspects—accuracy, prediction/detection time, and robustness. The paper suggests a hybrid approach which uses stateless and stateful features and then a 2-step approach to obtain better results. The dataset used here is described in Section 4. We first reimplement the 5 models presented in the paper, to serve as our baseline. We then analyze the data to identify characteristics typically associated with malicious queries and employ feature engineering and feature selection techniques and explore ensembling techniques. We monitor the prediction time along with metrics like precision, recall and f1 score to evaluate our models, validated with 5-fold cross validation. We test the validity and effectiveness of our ML models in the real-world, by testing the scheme on DNS data exfiltration queries from other tools like dns2tcp, dnscapy, iodine and tuns, as well as queries from the FrameworkPOS malware, which was notorious for stealing credit card information through DNS data exfiltration.

2. Related Work

Mahdavifar et al. (2021) have proposed a two-step method to detect DNS exfiltration making use of both stateless and stateful features to detect data exfiltration, using models such as Gaussian Naïve Bayes (GNB), Random Forest (RF), Multi-Layered Perceptron (MLP) Classifier, Support Vector Machine (SVM), and Logistic Regression. In the first step, the traffic is classified using a trained classifier based on stateless features. If the ratio of the suspicious samples in the packet window exceeds the set threshold, the traffic is re-analyzed using stateful features in the second layer. Based on their analysis, the authors observe that the Random Forest Classifier performs the best when compared to other machine learning for light attacks.

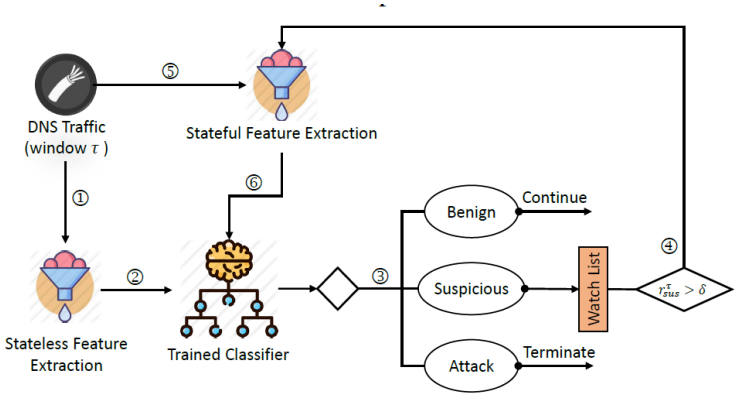

Figure 1. 2-Step model approach proposed by Mahdavifar et al. (2021)

Ahmed et al. (2019) discusses the use of only stateless attributes that can be computed in real-time without any prior knowledge. These included the total count of characters in FQDNs (Fully Qualified Domain Names), the count of uppercase characters, the count of numerical characters, the number of labels, and so on. An isolation forest anomaly detection model was trained using benign DNS queries. They tested their system on real-time Internet traffic of a medium-sized Research Institute and a large University. The reported prediction accuracy was around 98% during cross-validation. The results of this Machine Learning model show considerable efficiency when tested on a 10 Gbps real-time internet traffic stream injected with malicious queries processed using an online tool. Do et al. (2017) use stateful features (for example inter-arrival time between DNS query and response, size of the DNS query) to detect DNS tunneling over mobile networks. The authors use unsupervised learning models like K-Means clustering and One-Class Support Vector Machine (OCSVM). Their OCSVM, fine-tuned with the RBF kernel, was able to detect

the outliers correctly with an F1 score of 0.96. As the authors are unable to provide generalized results due to the small dataset, the scalability has not been evaluated. D'Angelo et al. (2022) propose a Convolutional Neural Network (CNN) approach to detect DNS tunneling. They convert the DNS queries extracted from the DNS Flow into bi-dimensional images and then utilize a CNN for image classification by utilizing the inbuilt features along with derived features which are inferred from the inbuilt features, for instance, the Response Entropy and Question Entropy. The authors state that it is the first implementation of the CNN for DNS tunneling detection. In our evaluation of the robustness and efficiency of the system, we do not consider CNN because of the speed and lack of interpretability.

3. Problem Statement

3.1. Issues with the 2-step model approach: The proposed 2-step approach by Mahdavifar et al. (2021) (described in section 2) achieves high accuracy metrics (precision, recall, and F1-score) on the provided experimental setup. In our work, we have replicated this experimental setup and obtained the accuracy metrics. However, when we plot the traffic and the ratio of malicious packets, as in Figure 2, we can see that one of the reasons that this approach works is that it classifies entire windows of data as malicious if the ratio of suspicious samples exceeds the threshold. The experiment is conducted over a period of 5 days, with a few hours of benign data followed by a few hours of malicious data on each day (except the first day, when only benign data is used). Since the attacks are spread out over long windows of either benign or malicious traffic, the only possible portion of the traffic where the overall accuracy can drop is the portion where the traffic switches from benign to malicious. Hence, the accuracy results from the 2-step model might be overly optimistic. Further, in the experimental setup, there is only a single client on the network. However, in a realistic scenario, there could be several clients on the network. As a result, most of the traffic is likely to be benign. Therefore, it would be highly unlikely that the ratio of suspicious data would cross a certain threshold because the traffic would most likely be dominated by benign data. Hence, not only are these results overly optimistic, but the experimental setup might also be too unrealistic for a deployment scenario.

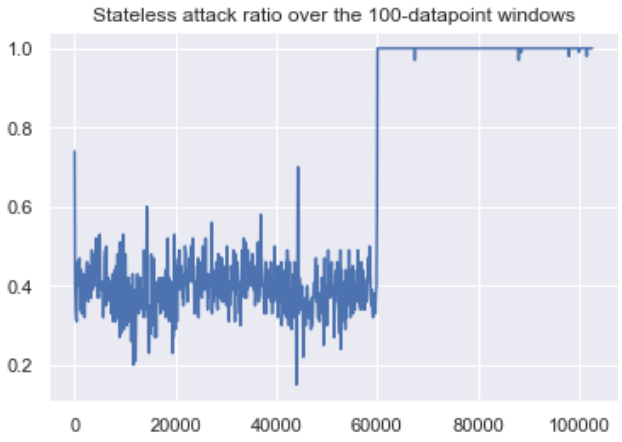

Figure 2. Illustration of the attack ratio over 100 datapoint windows within a subset of the data

3.2. Problem Formulation: Due to the above-mentioned flaws of the 2-step model, our work focuses on improving the detection on a packet basis, rather than a window basis. In other words, the focus is on improving and evaluating the performance of the stateless model in terms of accuracy, time taken, and robustness. All our baselines for this study are the stateless models proposed by the authors.

3.2.1. Accuracy: The ideal model should be able to classify every data packet as malicious or benign based on the features of the query. To evaluate the models, we use precision, recall, and F1-score.

3.2.2. Time: In order to minimize the amount of data being exfiltrated, the models need to be fast enough to detect exfiltration. As over 10,000 queries can pass through a network (Ahmed et al., 2019) per second, it is imperative that the feature extraction and exfiltration detection be completed in the span of less than a few seconds. All improvements we make are benchmarked against the original proposed models which use all the authors' proposed features.

3.2.3. Robustness: The data used in the authors' models and in our improved versions were created using the DNSExfiltrator tool. Despite the high accuracy, the proposed scheme needs to be robust enough to detect all kinds of DNS exfiltration when deployed. We test how well the models perform when dealing with DNS exfiltration queries generated from other DNS exfiltration tools and those generated from malware.

4. Dataset

The CIC-Bell-DNS-EXF-2021 (CIC-Bell-DNS-EXF 2021 | Datasets | Research | Canadian Institute for Cybersecurity | UNB, 2021) dataset consists of 270.8 MB DNS traffic including both benign traffic and malicious exfiltration traffic. To obtain the DNS traffic data, attacks were conducted by encoding and exfiltrating data belonging to various file types including audio, compressed, executables, image, text, and video using the DNSExfiltrator tool. Mahdavifar et al. (2021) were able to capture a total of 20.7MB, 147.6MB, and 102.5MB DNS packets for heavy, light, and benign traffic. To develop a real-world like generated dataset, they used distinct benign domains on each consecutive day of the heavy and light attack. They have developed a feature extractor to extract 30 features from DNS packets. The extractor was able to extract 14 stateless and 16 stateful features from the .pcap files. The feature extractor was not made publicly available, however.

Stateless features are features that do not use time independent characteristics of the hosts' DNS queries. Stateless features can be extracted from the individual DNS query packet (Ahmed et al., 2019). Stateful features, on the other hand, consider a set of queries within a particular time window, which inflicts a high computational cost on the DNS exfiltration detection system. After extracting the dataset, the authors obtained a final structured dataset of 323,698 heavy attack, 53,978 light attack, and 641,642 distinct benign samples. The dataset consists of .pcap and .csv files segregated into benign and heavy attacks. **Table 1** depicts the extracted feature with a brief description.

Feature Number	**Feature name**	**Description**	**State**
1	rr_type	The type of resource record	Stateful
2	rr_count	The total count of entries in each section	Stateful
3	rr_name length	The name length of resource record	Stateful
4	rr_name entropy	resource record name entropy	Stateful
5	rr_type frequency	Total number of packets of a given resource record type in each domain divided by the total number of packets in that domain.	Stateful
6	rr	The rate of A and AAAA records per domain in window τ	Stateful
7	distinct_ns	Number of distinct Name Servers	Stateful
8	a_records	Total number of IP addresses resolved in DNSDB	Stateful
9	unique_country	Distinct country names in each domain for window τ	Stateful
10	unique_asn	Unique ASN values in window τ	Stateful
11	unique_ttl	Unique TTL values in window τ	Stateful
12	distinct_ip	Unique IP values for a given domain in window τ	Stateful
13	distinct_domains	Distinct domains that share the same IP address that resolve to a given domain	Stateful
14	reverse_dns	Reverse DNS query results for a given domain in window τ	Stateful
15	ttl_mean	The average of Time to Live or TTL	Stateful
16	ttl_variance	The variance of Time to Live or TTL	Stateful
17	FQDN_count	Total Count of characters in FQDN	Stateless
18	subdomain length	Total Count of characters in subdomain	Stateless
19	upper	Total Count of uppercase characters	Stateless
20	lower	Total Count of lowercase characters	Stateless
21	numeric	Total Count of numerical characters	Stateless
22	entropy	Entropy of query name	Stateless
23	special	Number of special characters	Stateless
24	labels	Number of labels in a query	Stateless
25	labels_max	Maximum label length	Stateless

26	labels_average	Average label length	Stateless
27	longest_word	Longest meaningful word over domain length average	Stateless
28	sld	Second level domain	Stateless
29	len	Length of domain and subdomain	Stateless
30	subdomain	Shows whether a subdomain is present or not in the DNS Query	Stateless

Table 1. List of features in the dataset to detect DNS exfiltration.

5. Methodology

5.1. General approach and evaluation metrics:

When we build the stateless and stateful models to replicate Mahdavifar et al.'s (2021) approach, using separate models trained on the stateless and stateful features respectively. However, as mentioned before, we are only concerned with improving the stateless models. Unlike the paper's approach, we use stratified 5-fold cross validation to evaluate how the models perform, for all the improvements that we work on, in order to ensure that the results are more stable, and not merely a result of random chance. Additionally, unlike the paper's approach, we have included the calculation of prediction time within the pipeline.

5.2. Models

Mahdavifar et al. (2021) used five models: Logistic Regression, Random Forest, Gaussian Naïve Bayes (GNB) Classifier, Multi-Layered Perceptron (MLP) Classifier, and Support Vector Machines (SVM). We first reimplemented these models and then analyzed their performance metrics, and prediction times for only stateless features. Note that they have not discussed hyperparameter tuning and hence we did not use any specific parameters for the reimplementation step besides the default ones from the sklearn library. We attempted to improve the best models by reducing the number of complex features with the help of feature engineering and feature selection techniques.

5.3. EDA:

5.3.1. Goals and approach: The dataset was analyzed to understand the distribution of the different stateless features of the data. Figure 3 shows the distribution of the discrete numerical features, figure 4 shows the distribution of the continuous numerical features, and figure 5 shows the word cloud for the text-based features (longest_word and sld). Further, for the type of data being exfiltrated (audio, video, executables, etc.), the distribution of these features was also computed and can be seen in figure 10 in the appendix. During the analysis, regions of the input space that tended to be associated with attacks were identified and some of these insights were used for feature engineering.

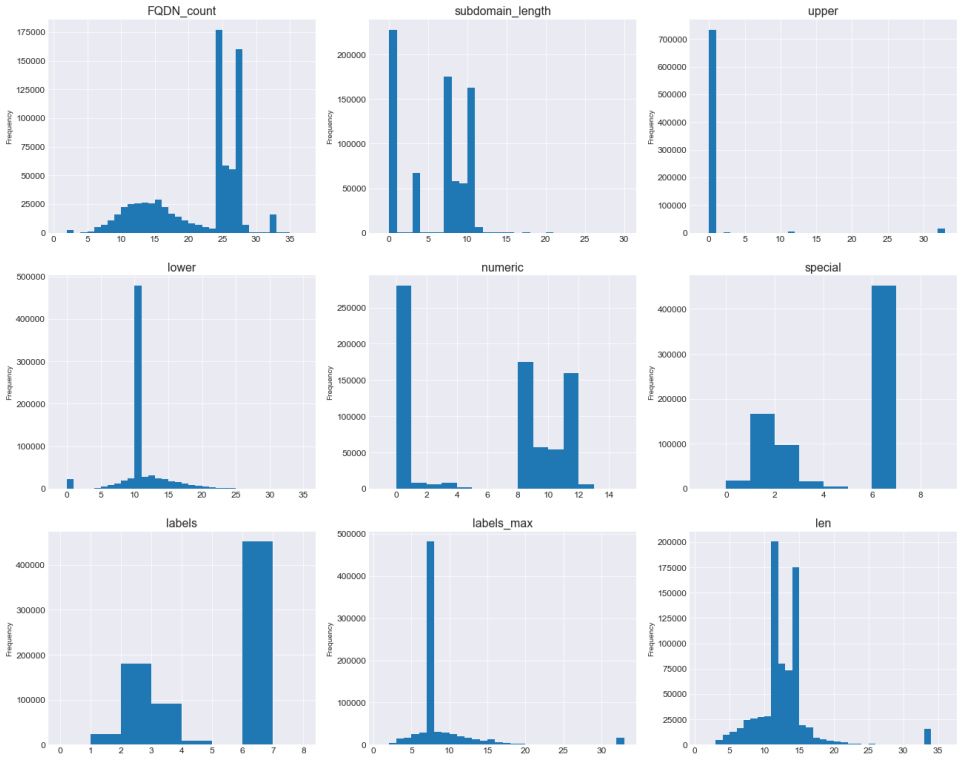

Figure 3: Distribution of the Discrete numerical features

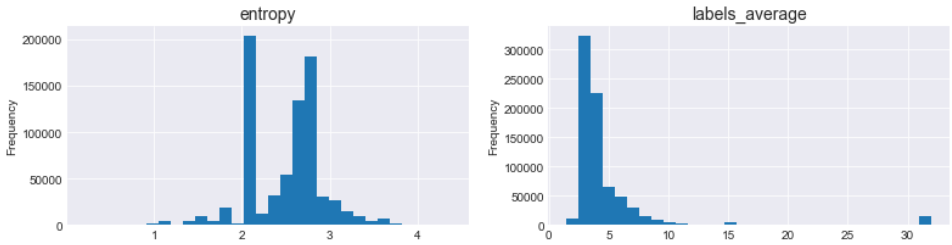

Figure 4: Distribution of the continuous numerical features

Figure 5: Word cloud for the longest meaningful word (the longest_word feature)

Figure 6: Word cloud for 'sld' feature:

5.3.2 Observations from the EDA:

- Shorter queries tend to be benign. Almost all (~97%) of queries with FQDN_count < 23 are benign instances.
- Queries without a subdomain or with a very short subdomain are almost always benign, queries with more lowercase letters (>10) are almost always benign, and queries without any numeric characters are almost always benign.
- Malicious queries are not guaranteed to have special characters. A huge proportion of malicious queries in the data had no special characters at all. Interestingly, when the number of special characters ranged from 1-4, the queries were almost always benign.

- Queries with a lower number of labels (<5) are almost always benign, queries that have a low maximum label length (<7) are almost always benign. Low (<2) and high entropy (>3) queries are almost always benign. (Note: This does not imply that moderate entropy queries are mostly malicious.)

- Most queries with a second-level domain (sld) of length greater than 3 tend to be benign, which suggests that the shorter sld's like 192, 239, etc. (which are part of IP addresses) are slighlty more likely to be associated with malicious queries. Also, slds with a mix of numbers and letters are almost always benign (~99%). Slds with only numeric characters are slightly more likely to be associated with malicious queries and those without any numeric characters are highly likely to be associated with benign queries (~95%).

- When the longest meaningful word found in the query has a length greater than 1, then the query is very likely to be benign, while single letter words found as the longest meaningful word (when there were no meaningful words found in the query) might be slightly more likely to be associated with malicious queries. Further, when the longest meaningful word is completely lowercase, it is highly likely to be benign (~98%)

5.4. Feature Engineering

Based on our analysis of the types of data associated with attacks, there are many possible features that could be used to help improve the models. In terms of the numerical features, it can be reasonably assumed that even simple machine learning models would be able to learn the patterns that were explained in section 5.3. However, for the text-based features, there are many patterns that might be missed by these models because the text inputs from the work by Mahdavifar et al. (2021) are one-hot encoded. Hence, we devise the following additional features to help provide the models with more meaningful information:

5.4.1. Numeric_3_character_sld: Within the second-level domain, if there are 3-digit numeric values like 192, 168, 224, etc., these are extracted. A default dummy value is used for cases where there is no 3-character numeric sld

5.4.2. sld_islower, sld_isupper, sld_isnumeric, sld_isalnum: These features are Boolean values that indicate whether the second-level domain is completely lowercase, uppercase, numeric, or alphanumeric (without special characters) respectively.

5.4.3. longest_word_islower,longest_word_isupper, longest_word_isnumeric, longest_word_isalnum: These features are Boolean values that indicate whether the longest meaningful word in the query is completely lowercase, uppercase, numeric, or alphanumeric respectively.

Another benefit of these features is that the time overhead to build text encodings is reduced, since we are extracting the most relevant information from the text, rather than encoding all the text. It is to be noted that not all these features were used. Feature selection (further described in section 5.5) was performed including the original and newly generated features, in order to determine the most relevant set of features.

Besides textual features, we attempted to read through the network traffic pcap data to extract more features. However, it appears that the Mahdavifar et al.'s (2021) extracted features are quite exhaustive as we were unable to extract any additional meaningful features, besides those that had already been extracted.

5.5. Feature Selection

The feature selection task was divided into 2 separate tasks to select the best stateless and stateful features with two different techniques, ANOVA and RFE.

5.5.1. Analysis of Variance (ANOVA)

ANOVA or Analysis of Variance uses F-Value scores to return the desired number of features in a dataset. We have used Scikit-learn library's SelectKBest method in which the score function has been set to 'f_classif' (F-classif scikit, n.d.) which is the ANOVA F-value score between each label or Feature for any given classification task. SelectKBest returns features with the highest scoring 'K' number of features. In our case the value of K was set to be 10 to select the 10 best stateless features from our list of original features combined with the features we had generated during the feature engineering step.

5.5.2. Recursive Feature Elimination (RFE)

RFE or Recursive Feature Elimination is a ranking based feature selection algorithm that selects features by recursively considering smaller sets of features (Feature selection. Scikit, n.d.). The importance of each feature is returned and the features with the least importance are pruned from the immediate set of features. This process occurs recursively until the desired number of features are achieved. We have used Scikit-learn library to implement the Recursive Feature Elimination process for the input data to select the 10 best stateless features.

After performing **feature engineering** with the text/categorical features we re-evaluated the base models to check any improvements in performance metrics and the prediction times.

5.6. Ensembling

Ensemble modeling is a process where multiple diverse models are created to predict an outcome, either by using many different modeling algorithms or using different training data sets. The ensemble model then aggregates the prediction of each base model and results in one final prediction for the unseen data. The motivation for using ensemble models is to reduce the generalization error of the prediction. As long as the base models are diverse and independent, the prediction error of the model decreases when the ensemble approach is used. The approach seeks the wisdom of crowds in making a prediction. Even though the ensemble model has multiple base models within the model, it acts and performs as a single model.

We segregated the data according to the file types because during the exploratory data analysis we found slight differences in feature distribution across the different file types (audio, video, image, text, etc.) Hence, we trained each of the individual models on data from only one file type. Then we created the following voting-based ensemble models to analyze the impact on prediction time and performance metrics.

a. Ensemble of logistic regressions
b. Ensemble of random forests
c. Ensemble of a mix of logistic regression and random forest

5.7. Queries from other DNS Exfiltration Tools

In order to validate how well our proposed approach works in "the real world", we needed to determine how robust it would be to other kinds of DNS exfiltration queries. To simulate this, we use queries generated from tools that were not used to generate the data used to train the model. As previously mentioned, the queries that were used by the models originated from the DNSExfiltrator tool, which uses the base64URL algorithm (Mahdavifar et al., 2021). For our robustness evaluation, we use DNS data exfiltration queries generated by the tools dns2tcp, dnscapy, iodine, and tuns, using the previous work by Ziza et al. (2022). These tools use different methods of encoding the exfiltrated data. The tool dns2tcp uses Base64 encoding on SSH traffic (Xia, 2016), dnscapy and iodine use SSH tunnels (Bienaimé, 2015; Madrigal, 2018), and tuns uses a more advanced method of only encapsulating the data within the CNAME records and modifying the MTU to be compatible with the DNS request data (Hinchliffe, 2019; Aiello et al., 2012). From these queries, we extracted all the features proposed by Mahdavifar et al.'s (2021), and then evaluated the two top baseline models, and our optimized versions of these models after performing feature engineering and selection. We also investigate if our learning schemes would be more robust to these queries if they were trained on such data, rather than using only queries from DNSExfiltrator, in order to determine if their performance drops are due to the learning scheme and features or due to the distribution of training data.

5.8. Malware Query Generation

To test the robustness of our models even further we wanted to see how well they would perform when faced with data exfiltration from actual malware. We selected the FrameworkPOS malware for the study.

FrameworkPOS is a malware that was used by hackers to steal credit card information from Point-of-Sale systems in the year 2014 from the American retailer Home Depot. FrameworkPOS uses DNS requests to exfiltrate the stolen data. For data obfuscation, this malware uses a special data encoding technique to encode credit card numbers into a DNS query (Rascagneres, P. 2016). Since we were not able to find a live version of this malware publicly available, we have replicated the behavior of the malware using the algorithm presented by Rascagneres (2016). In order to implement this algorithm, we have used the Malicious and Benign Websites dataset (Siddharta,

2021) to obtain malicious URLs and The Credit Card Dataset (Awasthi, 2018) to obtain dummy credit card numbers.

The encoding logic of the FrameworkPOS presented below was used to create dummy malware queries by embedding credit card data into the malicious URLs from the Malicious and Benign Websites dataset. 400 malware test queries were created by using the FrameworkPOS encoding logic. In execution, the FrameworkPOS malware creates encoded strings via XORing the byte sections in the loop via a sequence of XOR calls. The pseudo-code for the algorithm to encode credit card information into a DNS query is provided below:

```
For each bytes:
      a = byte XOR 0xAA
      b = a XOR 0x9B
      value = b XOR 0xC3
```

The following steps were undertaken to create the malware test data:

1. The Credit Card Data dataset was taken as an input to the Framework POS logic.
2. Framework POS logic encodes the credit card numbers using the above encoding algorithm. The results are shown in **Figure 7.**

Cardno	Encoded
8638540736318190	cac4c1cac7c6c2c5c1c4c1c3cac3cbc4
7106423970931510	c5c3c2c4c6c0c1cbc5c2cbc1c3c7c3c7
6492565582413530	c4c6cbc0c7c4c7c7cac0c6c3c1c7c1c2

Figure 7. Examples of encoded credit card numbers by the FrameworkPOS Logic.

3. Encoded credit cards are then added into the malware URLs taken from the Malicious and Benign Websites dataset.
4. The concatenated DNS queries contain encoded credit card data and can be used to test our models as shown in the examples in **Figure 8.**

```
c4c4cbc3c7c3c2c7c3c7c7c4c6c3c1c3.trtsport.cz
c3c6cac3c0c7c1c4c0c3c5cac5c7c6c5.trtsport.cz
c3c1c7c7c3c5c0cacac0c5c6cbc7cbc1.microencapsulation.readmyweather.com
```

Figure 8. Example of the generated dummy malware queries

6. Results and Discussion

6.1. Benchmarking the baseline models

	Precision	**Recall**	**F1-score**	**Prediction Time (in secs)**
Logistic Regression	0.611	0.999	0.758	3.033
Random Forest	0.611	0.999	0.758	193.85
GNB Classifier	0.611	0.971	0.750	5.143
MLPC Classifier	0.611	0.991	0.756	31.75
SVM	0.543	0.394	0.456	3.318

Table 2: Baseline models

The results of these baseline models did not look promising (Table 2) and appeared to have a lot of potential for improvement. (Note that we expect the baseline stateless models not to perform well, as Mahdavifar et al. (2021) used a combined 2-step approach that also used a stateful model in order to achieve the high reported accuracy. We had validated that we were able to achieve similar results when we reimplemented the 2-step approach.) Hence, we selected the top 2 (on the basis of speed and accuracy) models of this baseline reimplementation, i.e., Logistic Regression and Random Forest, to use for all our following approaches.

6.2. Feature Engineering and Feature Selection

By using the feature selection methods, we were able to reduce our total number of stateless from 23 (including the additional features generated during the feature engineering stage) to 10. We chose ANOVA as our final feature selection method as the accuracy drop of models trained with ANOVA's features was less than that of RFE's features. The models were again evaluated with the newly selected/reduced feature sets for both methods to finalize one technique for feature selection.

	Logistic Regression (Base Model)	Logistic Regression (After Feature Engineering)	Random Forest (Base Model)	Random Forest (After Feature Engineering)
Precision	0.611	0.611	0.611	0.611
Recall	0.999	0.994	0.999	0.998
F1-score	0.758	0.757	0.758	0.758
Prediction Time (in secs)	3.033	**1.609**	193.85	**39.082**

Table 3: Performance metrics after feature engineering

From **Table 3**, it is evident that we were able to improve prediction times using the combination of feature engineering and feature selection. For logistic regression, the prediction time was reduced to 53% of the initial prediction time, whereas, for the random forest, the prediction time was reduced to 20% of the initial prediction time.

In addition to this, feature importance values were also calculated to provide insights into the data, obtain the interpretability of the model, and form a basis for dimensionality reduction and feature selection to improve the efficiency and effectiveness of the model on the problem (Figure 9). From the graph, it is clear that the model pays special attention to the number of special characters and numeric characters. Among our crafted features, the the sld_islower feature significantly influences the prediction in the negative direction. Among the top 10 best features, 4 of them are from our crafted features, which indicates that these features were successful in simplifying the models. Even though there is not a significant reduction in accuracy, this feature engineering and selection step has improved the model interpretability and speed.

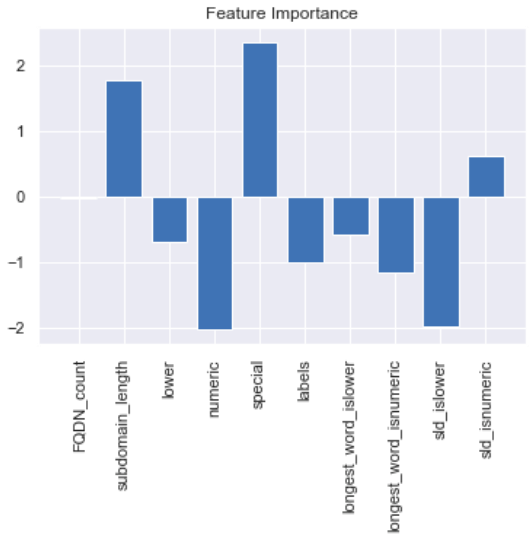

Figure 9: Feature Importance (LR) after doing feature engineering

6.3. Ensembling

The ensemble methods took a longer time to make predictions when compared to the base model, due to multiple models being used. At the same time, there were no significant improvements in the performance metrics, and hence we abandoned this approach.

6.4. Robustness Benchmarks:

6.4.1. Queries from Other DNS Exfiltration Tools

The usage of multiple tools like dns2tcp, dnscapy, iodine, tuns for generating exfiltrated data and testing robustness by passing that data through our optimized models (after feature engineering and feature selection) yielded the results in **Table 4**. It is clear from the table that Logistic Regression (LR) was able to perform well on some tools like iodine but not on others. Random Forest (RF) performed badly, not even able to make any predictions on any of the tool-generated data.

	dns2tcp		**dnscapy**		**iodine**		**tuns**	
Models	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**

LR Baseline	1.00	0.64	1.00	0.92	1.00	0.99	1.00	0.40
RF Baseline	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LR with optimized Features	1.00	0.65	1.00	0.92	1.00	0.99	1.00	0.40
RF with optimized Features	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 4. Performance metrics of models on exfiltrated data generated by other tools

A reason behind the bad performance of the Random Forest models can be **its inability to extrapolate in certain situations**. Logistic Regression models are capable of extrapolating well, i.e., they can perform well even when numerical features of test data are well outside the range of training data as they are developed on an arithmetic function (Wickramanayake, 2020). On the contrary, RF models might not perform well when the test data is far outside the range of the training data, and hence might fail to generalize.

When we train our optimized models on data from other queries, the performance of the models (validated with k-fold CV) is again high as shown in Table 5, which suggests that the learning scheme and extracted features are good enough to handle these queries, and that the main limiting factor was the diversity of data used in the training phase. Of course, since the list of DNS exfiltration tools used is not exhaustive, there are no guarantees that the learning scheme would generalize well to other types of DNS data exfiltration.

	Precision	Recall	F1-Score
Optimized LR	0.71	0.98	0.82
Optimized RF	0.70	0.96	0.81

Table 5. Optimized LR and RF model accuracy after training on queries from other DNS exfiltration tools

6.4.2. Malware (FrameworkPOS) Queries

The models were also tested on around 400 dummy malware queries containing encoded credit card information generated using the replication of the encoding algorithm of the FrameworkPOS malware. Table 6 clearly shows that our optimized models are not robust enough to handle queries from this malware. However, after training these models on data from other DNS exfiltration tools, they were able to generalize much better on such queries from this malware. Again, this suggests that perhaps the main limitation in the learning scheme might be the data, rather than the learning schemes. However, while this model performed well on FrameworkPOS, it might not necessarily generalize well on all DNS data exfiltration malware.

	LR baseline	**LR optimized**	**LR optimized after training on queries from other DNS exfiltration tools**	**RF baseline**	**RF optimized**	**RF optimized after training on queries from other DNS exfiltration tools**
Precision	0.00	1.00	1.00	0.00	0.00	1.00
Recall	0.00	0.22	1.00	0.00	0.00	1.00

Table 6: Performance of models on Dummy Malware Queries

7. Conclusion

In this study, we have attempted to improve upon the DNS data exfiltration detection techniques presented by Mahdavifar et al. (2021), by focusing on using stateless features to improve upon and evaluate the accuracy, time, and robustness of the detection techniques. Despite trying multiple techniques, we were unable to significantly improve upon the accuracy of the stateless models proposed by Mahdavifar et al. (2021), which could suggest that there is a limit to how well this task can be handled using only stateless query data. However, using a combination of feature engineering and feature selection, we were able to significantly improve upon the time taken for the detection, without a significant decrease in accuracy. This has several practical benefits, as it minimizes the amount of data that can be exfiltrated before the detection of such an attack. We have also investigated the robustness of this scheme to other types of DNS exfiltration attacks, including those from other tools and malware, and have found that the scheme is not very robust to such attacks. However, we have found that it could perhaps be the dataset, rather than the learning scheme that is the limiting factor, because the same models, when trained on data from a diverse set of exfiltration tools are more robust than the original models.

8. Future Work

For our future work, we can say that with the availability and inclusion of more representative data, the models can be trained better to be able to generalize well on unseen data. Also, the model could be benchmarked on live traffic to further evaluate the time and robustness in an even more realistic scenario. Transfer learning approaches could also be applied to further improve the robustness. The models could be further benchmarked on multiple different kinds of DNS exfiltration malware to have a better understanding of how well such learning schemes can handle real attacks from malware.

9. REFERENCES

Ahmed, J., Gharakheili, H. H., Raza, Q., Russell, C., & Sivaraman, V. (2019). Real-Time Detection of DNS Exfiltration and Tunneling from Enterprise Networks. 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 649–653.

Ahmed, J., Habibi Gharakheili, H., Raza, Q., Russell, C., & Sivaraman, V. (2020). Monitoring Enterprise DNS Queries for Detecting Data Exfiltration From Internal Hosts. IEEE Transactions on Network and Service Management, 17(1), 265–279. doi:10.1109/TNSM.2019.2940735

Aiello, M., Merlo, A., & Papaleo, G. (03 2012). Performance assessment and analysis of DNS tunneling tools. Logic Journal of IGPL, 21. doi:10.1093/jigpal/jzs029

Awasthi. (2018, August 24). Credit Card Data. Kaggle. https://www.kaggle.com/datasets/ananta/credit-card-data

Bienaimé. (2015). dnscapy. GitHub. https://github.com/FedericoCeratto/dnscapy

CIC-Bell-DNS-EXF 2021 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. (2021). https://www.unb.ca/cic/datasets/dns-exf-2021.html

D'Angelo, G., Castiglione, A., & Palmieri, F. (2022). DNS tunnels detection via DNS-images. Information Processing & Management, 59(3), 102930. doi:10.1016/j.ipm.2022.102930

Do, V. T., Engelstad, P., Feng, B., & van Do, T. (2017). Detection of DNS Tunneling in Mobile Networks Using Machine Learning. In K. Kim & N. Joukov (Eds.), Information Science and Applications 2017 (pp. 221–230). Singapore: Springer Singapore.

F-classif scikit (n.d.) sklearn.feature_selection.f_classif. Scikit-learn. Retrieved December 19, 2022, from https://scikit-learn/stable/modules/generated/sklearn.feature_selection.f_classif.html

Feature selection. scikit. (2020, May 25). Retrieved December 19, 2022, from https://scikit-learn.org/stable/modules/feature_selection.html#recursive-feature-elimination

Hinchliffe, A. (2019, March 15). DNS Tunneling: how DNS can be (ab)used by malicious actors. Unit 42. https://unit42.paloaltonetworks.com/dns-tunneling-how-dns-can-be-abused-by-malicious-actors/

Madrigal, C. (2015, October 18). DNS Tunneling with Iodine. https://calebmadrigal.com/dns-tunneling-with-iodine/

Mahdavifar, S., Hanafy Salem, A., Victor, P., Razavi, A. H., Garzon, M., Hellberg, N., & Lashkari, A. H. (2021). Lightweight Hybrid Detection of Data Exfiltration Using DNS Based on Machine Learning. 2021 the 11th

International Conference on Communication and Network Security, 80–86. Presented at the Weihai, China. doi:10.1145/3507509.3507520

Rascagneres, P. (2016, November 25). New FrameworkPOS variant exfiltrates data via DNS requests. G DATA | Trust in German Sicherheit. https://www.gdatasoftware.com/blog/2014/10/23942-new-frameworkpos-variant-exfiltrates-data-via-dns-requests

Sector. (2017). dns2tcp. GitHub. https://github.com/alex-sector/dns2tcp

Siddhartha. (2021, July 23). Malicious URLs dataset. Kaggle. https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset

Wickramanayake, B. (2021, December 14). Random Forest vs Logistic Regression. Medium. https://medium.com/@bemali_61284/random-forest-vs-logistic-regression-16c0c8e2484c

Xia, J. (2016, April 26). Analysis on Popular DNS Tunneling Tools. Infoblox Blog. https://blogs.infoblox.com/community/analysis-on-popular-dns-tunneling-tools/

Ziza, Kristijan; Vuletić, Pavle; Tadić, Predrag (2022), “DNS Exfiltration Dataset”, Mendeley Data, V2, doi: 10.17632/c4n7fckkz3.2

APPENDIX

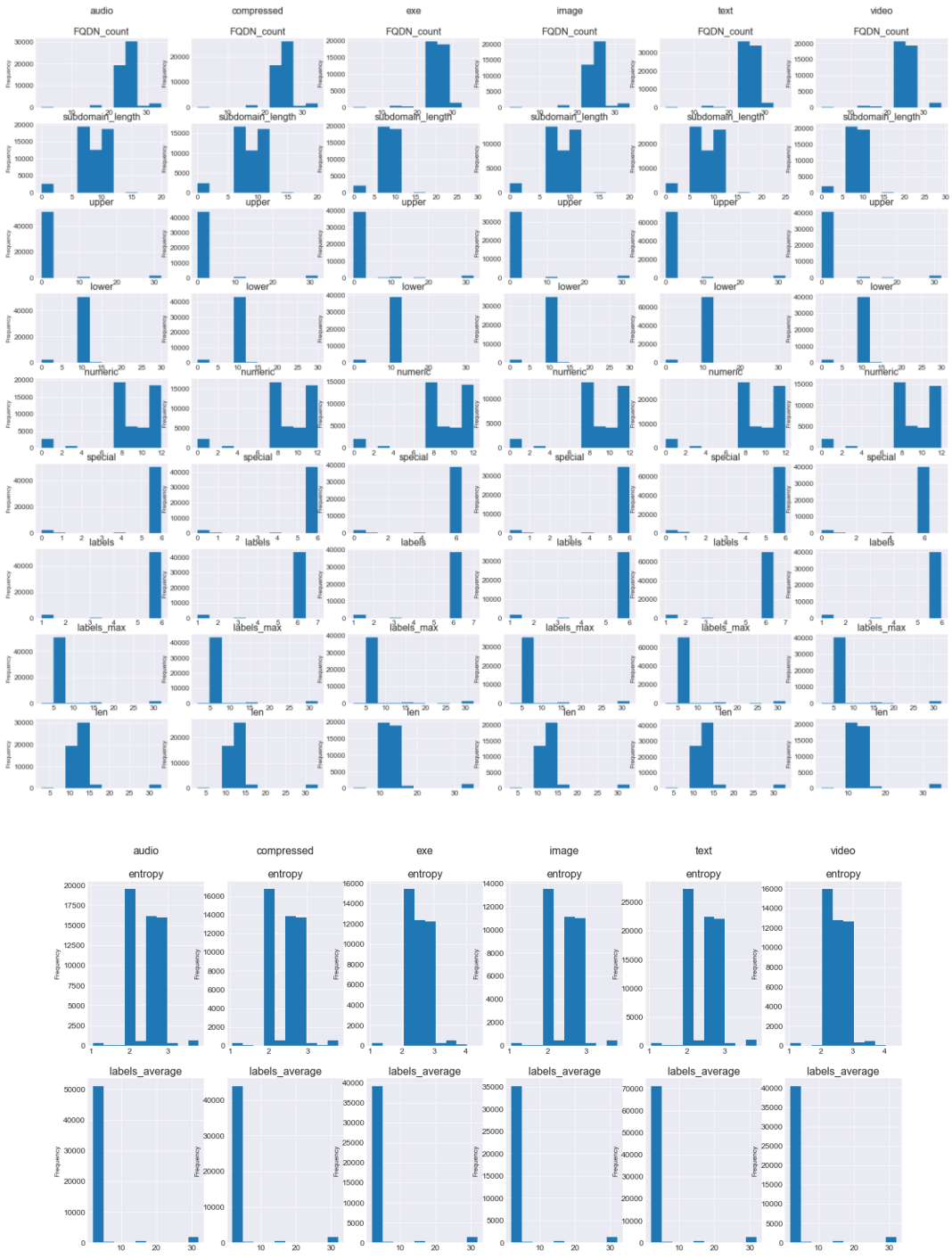

Figure 10. Feature Distributions among the different data types (audio, compressed, exe, image, text, video)