Lightweight Hybrid Detection of Data Exfiltration using DNS based on Machine Learning

Conference Paper · December 2021

DOI: 10.1145/3507509.3507520

7 authors, including:

Samaneh Mahdavifar
McGill University
13 PUBLICATIONS **395** CITATIONS
SEE PROFILE

Amir HOSSEIN Razavi
Bell Canada
25 PUBLICATIONS **865** CITATIONS
SEE PROFILE

Arash Habibi Lashkari
York University
177 PUBLICATIONS **5,917** CITATIONS
SEE PROFILE

Lightweight Hybrid Data Exfiltration using DNS based on Machine Learning

Samaneh Mahdavifar, Amgad Hanafy Salem, Princy Victor, Miguel Garzon, Amir H. Razavi, Arash Habibi Lashkari

Canadian Institute for Cybersecurity (CIC), University of New Brunswick (UNB);
Bell Canada (BCE), Cyber Threat Intelligence (CTI)
smahdavi@unb.ca, amgad.hanafy@unb.ca, princy.victor@unb.ca, miguel.garzon@bell.ca, amir.razavi@bell.ca, A.habibi.l@unb.ca

***Abstract*—Domain Name System (DNS) is a popular means to steal sensitive information from enterprise networks and maintain a covert tunnel for command and control communications with a malicious server. Due to the significant role of DNS services, enterprises often set the firewalls to let DNS traffic in, which encourages the adversaries to exfiltrate encoded data to a compromised server controlled by them. To detect low and slow data exfiltration and tunneling over DNS, in this paper, we develop a two-layered hybrid approach that uses a set of well-defined features. Because of the lightweight nature of the model in incorporating both stateless and stateful features, the proposed approach can be applied to resource-limited devices. Furthermore, our proposed model could be embedded into existing stateless-based detection systems to extend their capabilities in identifying advanced attacks. We release a large dataset of 270.8 MB DNS traffic generated by exfiltrating various file types ranging from small to big sizes. We leverage our developed feature extractor to extract 30 features from the DNS packets, resulting in a final structured dataset of 323,698 heavy attack, 53,978 light attack, and 641,642 distinct benign samples. The experimental analysis of utilizing several Machine Learning (ML) algorithms on our dataset shows the effectiveness of our hybrid detection system even in the existence of light DNS traffic.**
***keywords*— DNS, Data Exfiltration, Tunneling, Stateless, Stateful, Features, Machine Learning, Lightweight, Hybrid**

I. Introduction

Domain Name System (DNS) is a decentralized hierarchical system used to assign a unique Internet Protocol (IP) address to the domains names so that the browsers can access Internet resources. It works as follows: When a user tries to access a web address like "google.com", their web browser or application performs a DNS query by supplying the hostname to a DNS server. The DNS server takes the hostname and resolves it into a numeric IP address. A component called a DNS Resolver is responsible for checking if the hostname is available in the local cache, and if not, contacts a series of DNS Name Servers (DNS Root Server, Authoritative DNS Server), until eventually it receives the IP of the service the user is trying to reach and returns it to the browser at the client-side.

However, cybercriminals find it easy to take advantage of the DNS communication as the DNS traffic through UDP port 53 is not controlled in enterprises unlike services such as Hypertext Transfer Protocol (HTTP), email, or File Transfer Protocol (FTP). As a result, the cybercriminals purchase a domain for launching malicious activities such as ransomware, botnets, spam, click frauds, drive-by-downloads, Distributed Denial-of-Service (DDoS) attacks, protocol anomalies, tunneling, amplification, and reflection traffic, to name a few. According to the 2017 SANS data protection survey, among the 12% of security breaches in organizations, 48% involve data exfiltration attacks.

DNS data exfiltration is a type of attack where the data is exfiltrated from the computer through DNS protocol. In this attack, the attacker owns a server with malware and a domain that points to the server. During the exfiltration, the attacker makes the host query for the attacker domain and when the query is routed using the DNS resolver, a tunnel is created from the attacker to the target, which in turn permits the attacker to get the data, command and control the host, etc. As DNS is considered a noisy protocol, it is a tedious task to detect the data exfiltration attacks.

In this paper, we propose a hybrid approach for detecting DNS data exfiltration attacks that employs both stateless and stateful DNS features to efficiently identify malicious DNS traffic. This lightweight approach could be extensible on the detection systems with stateless features only and can be deployed on resource-constrained devices. To show the efficacy of our proposed model, we analyze it with several Machine Learning (ML) models, namely Gaussian Naive Bayes (GNB), Random Forest (RF), Multi-layer Perceptron (MLP), Support Vector Machine (SVM), and Logistic Regression (LR). The experimental results show that our proposed approach can remarkably distinguish the DNS data exfiltration attack even for light traffic flowing between the victim machine to the attacker's server. RF outperforms other classifiers with an accuracy of 99.94% and 99.97% for light attack and heavy attack, respectively. Specifically, our contributions can be summarized as follows:

- We develop 30 stateful and stateless features to detect data-stealing over DNS protocol. The feature types are used in a two-layered architecture where in the first layer the traffic is classified using a trained classifier based on stateless features. If the ratio of the suspicious samples in the packet window τ exceeds threshold δ, the traffic is re-analyzed using stateful features in the second layer.
- We generate and release a large dataset for detecting

DNS data exfiltration attacks, including 20.7MB light attack, 147.6MB heavy attack, and 102.5MB distinct benign DNS packets and a final structured CSV files of 323,698 heavy attack, 53,978 light attack, and 641,642 distinct benign samples. Each heavy and light attack category comprises of six diverse file types, including audio, compressed, exe., image, text, and video.

- Our developed lightweight hybrid approach could be easily deployed on resource-constrained devices. In real-world scenarios, the vast majority of the DNS traffic contains benign load. In this case, our proposed approach would be so efficient in working based on the stateless features only and liberates us from the tedious task of extracting stateful features.
- Our proposed approach could be embedded as a module to existing DNS data exfiltration detection systems that are based on stateless features only and could be upgraded with stateful detection.

The rest of the paper is organized as follows. Section II reviews background research. Related work is discussed in Section III. We describe the extracted features in Section IV. In Section V, our proposed hybrid lightweight approach is explained. Section VI discusses the generated dataset and the experimental analysis. Finally, Section VII concludes the paper and presents some future work.

II. Background

A. Data Exfiltration Using DNS

In data exfiltration using a DNS attack, the DNS system is exploited to transfer data from a machine that has already been compromised by a malicious actor to an external server controlled by an adversary. The transfer of data can be manual by someone with physical access to the computer or automated which is carried out through malware over a network [1]. In a simplified version, the malware on the compromised machine employs DNS tunnelling may typically have code running on a computer that periodically polls for commands to run, and responds with the output of those commands encoded as a series of A record look-ups to an attacker controlled domain that are reconstructed server-side. Consider a compromised machine has been infiltrated by a malware that aims to send out certain information to a malicious server with domain (malicioussite.com). The attack is conducted in the following scenarios:

- The adversary encodes the data using techniques usually base64URL.
- The malware can then craft a DNS query with the encoded data as the subdomain, e.g., aGVsbG8gaG93IGFyZSB5b3Vu.malicioussite.com.
- The query is first looked up in the cache and then local and public DNS servers. However, since the subdomain does not exist, the query is forwarded to the dedicated name server of malicioussite.com to resolve the IP address [2].
- When the malicious name server receives the query, the adversary will extract the Third Level Domain (3LD) which is aGVsbG8gaG93IGFyZSB5b3Vu, and decode the information.
- The name server can then choose an appropriate reply for the query to make it look benign.

Figure 1 depicts the exfiltration of details, such as Name, SSN, and date of birth of an individual through DNS queries. In this example, the data is divided into chunks and attached to the attacker domain fooserver.com to pass it through the firewall, as firewall does not block DNS queries. The query is first looked into the local DNS server and since it does not resolve the query, it will be forwarded to attacker server, as shown in the figure. The data is then extracted from the server and a response that almost satisfies the query will be sent back to the victim machine.

III. Related Work

The use of the DNS protocol for data exfiltration was first discussed in 1998. Twenty years on, this covert transmission method has become more sophisticated as malicious actors adapt to evade detection techniques. Enterprise networks constantly face the threat of valuable and sensitive data being stolen by cyber-attackers. Adversaries are progressively exploiting the DNS for sending out encoded private information from compromised hosts, as well as maintaining a covert command and control communications channel for automated malware. Meanwhile, enterprise firewalls are configured to allow DNS traffic on UDP port 53 pass through since DNS is such a crucial service for all applications. On the other hand, deeply inspection of DNS packets to identify covert channel will impact firewall forwarding performance, therefore enterprises usually disable enhanced DNS protection. In this section, we review a series of work has been done to conduct exfiltration over DNS and/or detect the attack.

DNS-based data exfiltration may be accomplished without additional software or user privileges by quickly implementing client-side Javascript in a text editor that can be launched in the system's browser. To exfiltrate documents over DNS using JavaScript, the file must first be read from the local system and turned into a binary string, optionally compressing or encrypting it in the process. The data must then be encoded using the allowable DNS characters specified in RFC 1035. We can employ base32 or base64 encodings. The final data representation may then be broken into segments that can be exfiltrated in separate DNS queries. Born [3] explores different methods of using a browser's JavaScript engine to exfiltrate documents over the DNS protocol without sending less covert HTTP requests. It has been shown how we can provide a bi-directional covert channel between the client and server.

In [4], a new way of data exfiltration is proposed that uses system resources instead of a malware. This type of attack is specifically helpful to Advanced Persistent Attack (APT) threat actors who reside in a host for along period of time without being detected and seek the best way to exfiltrate data. System resources comprise Remote Desktop Protocol (RDP), PoweShell, Windows accessibility backdoor, and DNS tunneling. The authors then recommend some mitigation techniques

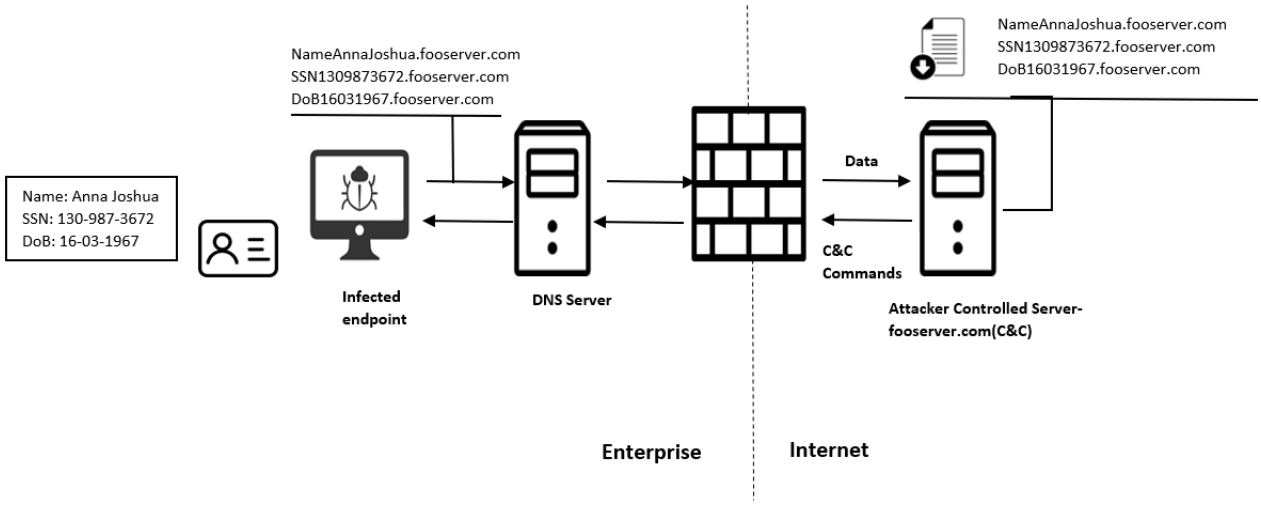

Fig. 1: Data Exfiltration via DNS queries

based on analyzing traffic captured from secure DNS tunnels over the network.

Das et al. [2] develop ML models to detect data exfiltration and tunneling over DNS. They experiment the effectiveness of their model by running a real-world malware sample that was used in targeted attack against major US financial institutions and trigger an alert. In the end, they discussed potential methods that could be used to bypass their detection. In another work [5], a real-time DNS data exfiltration detection mechanism is proposed using isolation Forest ML algorithm. In the model, network traffic from two enterprise networks are collected to identify the anomalous DNS queries and based on the findings, the scheme is implemented in a live traffic.

ML is also used in [6], where the DNS-based data exfiltration is detected by injecting mobile agents in the network for monitoring the network traffic. The agents then use the Zeek tool for periodical feature vector creation and the features thus extracted are used for the exfiltration detection using C5.0 and decision tree machine learning algorithms. If malicious incidents are reported, the mobile agents move to the malicious source and kill the process.

In paper [7], DNSxD is proposed for data exfiltration analysis, detection, and mitigation built on the Software-Defined Network (SDN) architecture. They analyzed exisitng data exfiltration methods and various attack tools to generate a feature-set for detecting data exfiltration using DNS. However, the work had some limitations, and to address these issues, another architecture DNSxP is proposed in [8]. In this model, DNS request is passed through four different modules, such as packet classifier, packet mirror, ONOS-based DNS monitoring application, and pDNS for determining whether the request needs to be dropped or not. In the packet classifier module, DNS requests are classified as benign, suspicious, or malicious, before passing on to the P4 switch. The suspicious and benign packets are then fed to the P4 switch in the packet mirror module, which decides whether the packet needs to be mirrored or not. Once the decision is made, the benign packets are directly sent to the internal DNS server, and suspicious packets are sent to the internal DNS server after mirroring the traffic to the controller. Using an ONOS-based monitoring application, the controller extracts the domain and analyzes the packet based on the threshold of features, and updates the blacklist if it is malicious. Finally, the pDNS module drops the blacklisted packets and forwards the normal packets to an external DNS Server.

Nadler et al. [9] propose a method based on anomaly detection, which is capable of detecting both DNS tunneling and DNS data exfiltration malware. The proposed method inspects recent history of DNS traffic logs rather than a short time window which may be inefficient for low throughput exfiltration. Periodically, a feature extraction phase is applied to the collected logs of each domain. Then, the domain is classified using the extracted features to determine if the domain is used for data exchange

Recently, Ahmed et al. [10] propose a real-time mechanism for detecting exfiltration and tunneling of data over DNS in the network edge. They have analyzed DNS traffics from two organizations and identified stateless attributes of DNS messages, such as length, entropy, dots, numerics, uppercase characters, and the number of labels, that can distinguish malicious from legitimate queries. Then, they develop and tune an ML algorithm to detect anomalous DNS queries. Finally, they implement their scheme on live 10 Gbps traffic streams from the network borders of the two organizations, inject more than three million malicious DNS queries generated using two customized exfiltration tools and show that their scheme is able to identify such malicious activity.

Another study is a patent to detect and block data transfer using DNS protocol [11]. Their proposed method comprises a DNS that monitors the queries raised by a DNS client. The DNS identifies at least one DNS query whose fully Qualified Domain Name (FQDN) is not in the cache of the DNS. Then the DNS detects an exfiltration event based on a summation of size of the DNS queries and/or DNS responses. Based on

the detection of the exfiltration event, any further DNS queries are then blocked from the DNS client.

One of the challenges of DNS data exfiltration and tunneling detection, is that they are vulnerable against feature obfuscation. To address this issue, Ishikura et al. [12] leverages the fact that in DNS data exfiltration and tunneling, upon sending a request from the malicious client to the server, a cache miss always happens on the DNS server. So, they propose two types of features based on cache properties, i.e. Cache Hit Ratio (CHR) and Access Hit Ratio (AHR) to detect DNS tunneling. They showed that AHR could characterize DNS tunneling traffic.

Overall, the research studies proved that the stateful as well as stateless features are highly efficient in DNS Data exfiltration detection and to the best of our knowledge, no literature has considered them both in their works. It is also interesting to notice that the existing works focus on either heavy attacks or light attacks during the DNS data exfiltration detection. Unlike others, in this paper, a model is proposed where heavy and light attacks are detected by using the aforementioned stateless and stateful features.

IV. Proposed Features

The DNS message consists of four sections: question, answer, authority and additional. The question and answer sections are the primary candidates to be exploit by the DNS data exfiltration and tunneling software. The question section contains the name being queried, or encoded binary data being transferred to adversarial DNS server within queried name. The answer section encloses requested Resource Record (RR), or arbitrary data being transferred from the adversary DNS server back to the client. Authority and additional sections inherit format of the answer section, moreover can be exploited by the server to deliver commands and data to the infected clients [13].

Table I shows the features extracted to detect data exfiltration over DNS. Generally, the features are divided into two large groups: stateless and stateful. Stateless features are independent of time-series characteristics of queried domains or hosts' DNS activity and can be derived from individual DNS query packets. This reduces the overhead in computing these attributes in real-time [10]. In contrast, stateful features consider a range of queries in a time window and thus inflict high computational cost on the detection system. However, stateful detection allows scanning DNS logs in a long period of time and therefore, can deal with low and slow DNS attacks.

V. Proposed Hybrid Lightweight Approach

In this section, we explain an overview of our proposed approach to determine whether a DNS query is normal or attack. We aim to design a lightweight approach, so it could be deployed in resource-constrained devices. As shown in Figure 2, we provide a two-layered approach in which the stateless features are extracted from the incoming DNS traffic in window τ (window of packets), and then the structured data goes through a trained classifier. The classifier output probability is then divided into three bins, i.e., $[0-0.4[$, $[0.4-0.7[$, $[0.7-1]$ to help the classifier score each input samples in window τ as benign, suspicious, or malicious. If the ratio of the suspicious samples in window τ, i.e. r^{τ}_{sus}, exceeds the threshold δ, the whole traffic window is re-analyzed using stateful features to let the trained classifier on stateful features decide about the whole window τ. Otherwise, the input sample is either identified as benign for which the DNS traffic keeps on flowing, or is detected as attack for which we terminate the DNS traffic.

Stateful features could be leveraged as a supplementary material to consider the DNS traffic in case a noticeable portion of the packets in a packet window are suspicious, which could be further investigated. Using stateful features, the classifier determines the maliciousness degree of the whole window and not the individual packet.

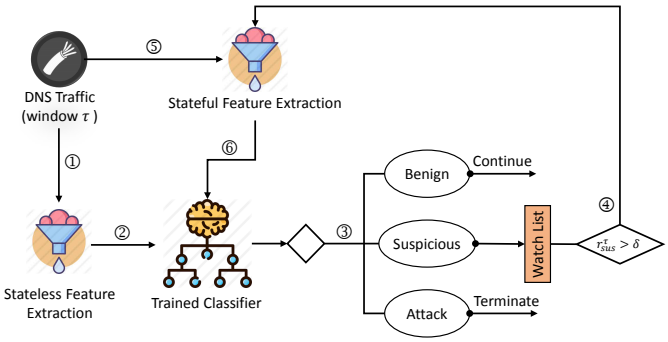

Fig. 2: Proposed approach overview

VI. Experimental Analysis

A. Testbed

Figure 3 explains how data exfiltration attack using DNS is run on the Canadian Institute for Cybersecurity (CIC) testbed. The data is encoded on the client-side (victim's side) and piggy-backed on DNS requests to the DNS server set as the name server of the attacker's machine. Practically, the server-side (attacker's side) acts as a malicious DNS server and receives the encoded file. The file is then decoded to see the content.

We use DNSExfiltrator tool, publicly available on github [18], which helps us for conveying a file over a DNS request covert channel. We also registered a domain name, namely cicresearch.ca and set the NS record for that domain to point to the attacker's server that will run the server-side script. In the DNSExfiltrator tool, the encoding algorithm and the throttling time are set to base64URL and 500 ms, respectively. The maximum size in bytes for each DNS request is set to the default value (255 bytes) and the maximum size in chars for each DNS request label (subdomain) is set to default (63 characters).

B. Dataset

We used DNS active data collection method for collecting DNS data. We collected benign samples from Alexa top 1-million domains. For collecting DNS data exfiltration attack

TABLE I: List of DNS features for detecting DNS data exfiltration

Feature	Feature name	Description	State
F1	rr_type	The type of resource record [13], e.g., A, TXT, MX, ...	stateful
F2	rr_count	The count of entries in each section question, answer, authority, and additional [14]	stateful
F3	rr_name_length	The resource record name length [14]	stateful
F4	$rr\_name\_entropy$	The entropy of resource record name [14]	stateful
F5	$rr\_type\_frequency$	Number of packets of a given resource record type for a given domain over the total number of packets for that domain (where qtype is A, AAA, CNAME, MX, NAPTR, NS, NULL, SOA, TXT, STAR, SRV, and PTR) [14]; example feature names: A_Frequency, TXT_Frequency, ...	stateful
F6	rr	Distribution of A and AAAA resource records [9], i.e., the rate of A and AAAA records per domain in window τ	stateful
F7	$distinct\_ns$	Number of distinct NS records [15], i.e., the total number of name servers (NS) resolved in DNSDB	stateful
F8	$a\_records$	Number of distinct A records [15], i.e., the total number of IP addresses resolved in DNSDB	stateful
F9	$unique\_country$	Distinct country names for a given domain in window τ [16]	stateful
F10	$unique\_asn$	Distinct ASN values in window τ	stateful
F11	$unique\_ttl$	Distinct TTL values in window τ	stateful
F12	$distinct\_ip$	Distinct IP values for a given domain in window τ [16]	stateful
F13	$distinct\_domains$	Distinct domains that share the same IP address that resolve to a given domain [16] in window τ	stateful
F14	$reverse\_dns$	Reverse DNS query results for a given domain in window τ [16]	stateful
F15	ttl_mean	The average of TTL in window τ	stateful
F16	$ttl\_variance$	The variance of TTL in window τ	stateful
F17	$FQDN\_count$	Total count of characters in FQDN [10], [17]	stateless
F18	$subdomain\_length$	Count of characters in subdomain [10]	stateless
F19	$upper$	Count of uppercase characters [2], [10]	stateless
F20	$lower$	Count of lowercase characters [2]	stateless
F21	$numeric$	Count of numerical characters [2], [10]	stateless
F22	$entropy$	Entropy of query name [2], [10], [17]: $H(X) = -\sum_{k=1}^{N} P(x_k) log_2 P(x_k)$, X =query name, N =total number of unique characters, $P(x_k)$ =the probability of the k-th symbol	stateless
F23	$special$	Number of special characters [2]; special characters such as dash, underscore, equal sign, space, tab	stateless
F24	$labels$	Number of labels [2], [10]; e.g., in the query name "www.scholar.google.com", there are four labels separated by dots	stateless
F25	$labels\_max$	Maximum label length [10]	stateless
F26	$labels\_average$	Average label length [10]	stateless
F27	$longest\_word$	Longest meaningful word over domain length average [9]	stateless
F28	sld	Second level domain	stateless
F29	len	Length of domain and subdomain	stateless
F30	$subdomain$	Whether the domain has subdomain or not	stateless

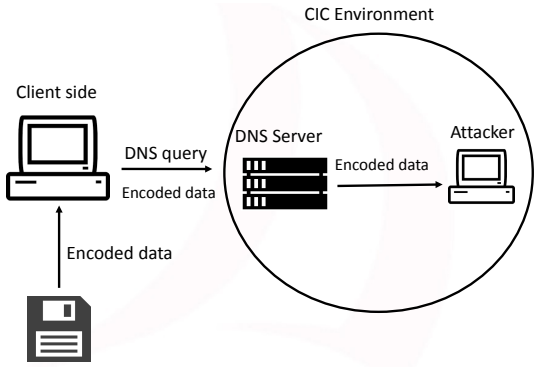

Fig. 3: DNS Exfiltration testbed

traffic, we conducted the attack in two categories of light file attack and heavy file attack in five consecutive days. There are six file types in each heavy and light category including, audio, compressed, exe., image, text, and video. The size of the light file category ranges from 15KB to 924KB while the size of heavy files ranges from 4.5 MB to 26.9MB. For capturing the benign traffic, we sent HTTP requests to the collected domains' web server using a Python script and dump the packets with an OK response. To acquire a real-world generated dataset, we use distinct benign domains on each consecutive day.

The attack scenario is as follows:

- First day (Benign)
  - Friday 20th November: 9:59 am-12:57am
  - Benign: 9:59 am-12:57am (35,636 domains)
- Second day (Light Attack)
  - Saturday 21st November
  - Benign: 10:18 am-2:00pm (9,956 domains)
  - Attack
    - Audio: 3:13 pm-3:50 pm

    - Compressed: 6:09 pm-7:49 pm
    - Exe.: 7:52 pm-8:48 pm
    - Image: 6:46 pm-9:51 pm
    - Text: 10:21 pm-10:43pm
    - Video: 10:56-11:37 pm
- Third Day (Heavy Attack)
  - Sunday 22nd November
  - Benign: 6:53 am-10:43 am (9,956 domains)
  - Attack
    - Audio: 10:52 am-4:17 pm
    - Compressed: 4:46 pm-9:07 pm
- Fourth Day (Heavy Attack)
  - Monday 23rd November
  - Benign: 11:06 am-2:21 pm (8,403 domains)
  - Attack
    - Image: 2:27 pm-8:24 pm
    - Text: 8:28 pm-12:15 am
- Fifth Day (Heavy Attack)
  - Tuesday 24th November
  - Benign: 8:09 am-12:53 pm (11,704 domains)
  - Attack
    - Video: 1:00 pm-7:16
    - Exe.: 7:18 pm-12:58 pm

All the benign and attack traffic were captured using tcpdum on the victim's side and labeled according to their timestamps. We captured a total of 20.7MB, 147.6MB, and 102.5MB DNS packets for heavy, light, and benign traffic.

We then applied our developed DNS feature extractor package to extract 14 stateless and 16 stateful features from all .pcap files. The benign/attack ratio for each pair of heavy-stateful, heavy stateless, light-stateful, and light-stateless is $60/40\%$.

Table II shows the statistics of the captured DNS packets and final structured CSV files in three categories of benign, light attack, and heavy attack. Each row in our structured dataset depicts a timestamped DNS packet along with the 30 extracted features. To keep the benign/attack ratio, we injected the benign packets on the first day to benign packets on the light attack day, i.e., light-benign, and similarly to the benign packets on the three heavy attack days, i.e., heavy-benign.

TABLE II: Statistics of the dataset

Category	#Stateful	#Stateless	#DNS Packets
Heavy Attack	72,028	251,670	147.6MB
Heavy-Benign	156,014	402,767	90.3 MB
Light Attack	11,295	42,683	20.7MB
Light-Benign	109,766	281,164	62.4MB

C. Experimental Results

To evaluate the effectiveness of our proposed lightweight approach, we follow the model in Section V where in the first layer, the DNS traffic goes through a stateless detection and being conditioned on the ratio of the suspicious samples, a stateful filtering is applied to the DNS incoming traffic. We conduct the experiments for DNS data exfiltration attack with heavy and light traffic and compare the results.

1) Sanitization and parameter setting: In the preprocessing step, we remove the timestamps from the features to prevent ML overfitting problems. We sanitize the data by replacing nan values with zero. Furthermore, we encode the stateful and stateless categorical features. The stateful categorical features include, rr_type, $distinct\_ip$, $unique\_country$, $unique\_asn$, $distinct\_domains$, $reverse\_dns$, and the stateless categorical features consist of $longest\_word$ and sld. We also substitute the $unique\_ttl$ lists with the average of the ttl values in each list.

For the parameter setting, the window size τ is set to 100 packets and the sliding window step s is set to 100. We choose a small value for τ to avoid a high false-positive rate. Based on Figure 2, if the stateless classifier detects even one packet malicious, we dump the whole packet window. Therefore, setting the window size to a fairly large value might result in shutting down a noticeable portion of the benign DNS traffic which is not desirable in a real-world situation. The threshold for the ratio of the suspicious samples, i.e., δ, is also considered as 0.4.

2) Analysis: We develop four classification algorithms using Scikit-learn library [19] in Python including GNB, RF, MLP, SVM, and LR. We set the train-test split ratio to (70%-30%) and shuffle the entire dataset before splitting.

The classification performance of the four algorithms on the heavy and light attacks is shown in Table III. The results in Table III show that RF highly outperforms other classifiers for light attacks. It increases the ACC of MLP by 4.62% and LR by 4.81%. In contrast, for the heavy attacks, RF and MLP achieve remarkably better performance with a negligible difference (99.97% and 99.84% accuracy, respectively). Surprisingly, light attack detection using SVM works better than the heavy attack, acquiring 88.6% accuracy and 83.25% F1_Score. The same situation exists for GNB and LR, however, the pair-wise gap between the performance of these two models is not much flagrant as SVM does. The reason for MLP superiority in heavy attack detection versus light attack detection is the higher number of samples that serves heavy attack detection when training a neural network.

TABLE III: Classification performance (%) of common ML algorithms for detecting data exfiltration over DNS

ML Models	Light Attack			Heavy Attack		
	Precision	F1_Score	Accuracy	Precision	F1_Score	Accuracy
GNB	91.41	74.94	69.43	83.46	71.24	71.01
RF	99.94	99.94	99.94	99.97	99.97	99.97
MLP	91.13	93.12	95.32	99.84	99.84	99.84
SVM	78.53	83.25	88.60	65.31	64.90	71.36
LR	92.97	93.39	95.13	92.07	89.59	89.22

The overall classification results demonstrate the effectiveness of our proposed approach in identifying even the light DNS traffic.

VII. Conclusion

In enterprise networks, handling the threat of DNS data exfiltration that aims to collect sensitive data is a tedious task.

In this work, a two-layered lightweight approach is employed to detect heavy and light DNS data exfiltration attacks in enterprise networks. Rather than considering either stateful or stateless features, we make use of both features in our model that leads to an enhanced performance. The efficacy of the model is evaluated using the dataset in four ML algorithms such as GNB, RF, MLP, SVM, and LR. Experimental results proved that RF outperforms other algorithms in detecting light and heavy attacks. A key advantage of the proposed lightweight strategy is the capability to detect DNS data exfiltration attacks in resource-constrained devices in a better manner.

For our future work, studies will be conducted on making use of the model in classifying data exfiltration attack in live DNS network traffic.

REFERENCES

[1] "What is a data exfiltration?" https://www.infoblox.com/glossary/data-exfiltration/, accessed June 2020.

[2] A. Das, M.-Y. Shen, M. Shashanka, and J. Wang, "Detection of exfiltration and tunneling over dns," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2017, pp. 737–742.

[3] K. Born, "Browser-based covert data exfiltration," *arXiv preprint arXiv:1004.4357*, 2010.

[4] A. Zimba and M. Chishimba, "Exploitation of dns tunneling for optimization of data exfiltration in malware-free apt intrusions," *Zambia ICT Journal*, vol. 1, no. 1, pp. 51–56, 2017.

[5] J. Ahmed, H. H. Gharakheili, Q. Raza, C. Russell, and V. Sivaraman, "Real-time detection of dns exfiltration and tunneling from enterprise networks," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 649–653.

[6] T. Aurisch, P. C. Chacón, and A. Jacke, "Mobile cyber defense agents for low throughput dns-based data exfiltration detection in military networks," in *2021 International Conference on Military Communication and Information Systems (ICMCIS)*, 2021, pp. 1–8.

[7] J. Steadman and S. Scott-Hayward, "Dnsxd: Detecting data exfiltration over dns," in *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2018, pp. 1–6.

[8] ——, "Dnsxp: Enhancing data exfiltration protection through data plane programmability," *Computer Networks*, p. 108174, 2021.

[9] 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.

[10] J. Ahmed, H. H. Gharakheili, Q. Raza, C. Russell, and V. Sivaraman, "Monitoring enterprise dns queries for detecting data exfiltration from internal hosts," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 265–279, 2019.

[11] S. Galliano and J.-y. Bisiaux, "Method and system for detecting and blocking data transfer using dns protocol," May 7 2020, uS Patent App. 16/248,337.

[12] N. Ishikura, D. Kondo, I. Iordanov, V. Vassiliades, and H. Tode, "Cache-property-aware features for dns tunneling detection," in *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. IEEE, 2020, pp. 216–220.

[13] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[14] Y. Bubnov, "Dns tunneling detection using feedforward neural network," *European Journal of Engineering and Technology Research*, vol. 3, no. 11, pp. 16–19, 2018.

[15] Z. Liu, Y. Zeng, P. Zhang, J. Xue, J. Zhang, and J. Liu, "An imbalanced malicious domains detection method based on passive dns traffic analysis," *Security and Communication Networks*, vol. 2018, 2018.

[16] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "Exposure: A passive dns analysis service to detect and report malicious domains," *ACM Transactions on Information and System Security (TISSEC)*, vol. 16, no. 4, pp. 1–28, 2014.

[17] I. Homem and P. Papapetrou, "Harnessing predictive models for assisting network forensic investigations of dns tunnels," 2017.

[18] "Dnsexfiltrator," Nov. accessed Nov. 2020. [Online]. Available: https://github.com/Arno0x/DNSExfiltrator

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.