A security model for DNS tunnel detection on cloud platform

Lorena de Souza Bezerra Borges [*†], Robson de Oliveira Albuquerque [‡†], Rafael Timóteo de Sousa Júnior [§†]
Professional Graduate Program in Electrical Engineering – PPEE
Electrical Engineering Departament, Faculty of Tecnology, University of Brasília – UnB[†]
Email: lorenabez@gmail.com*, robson@redes.unb.br‡, desousa@unb.br§

2022 Workshop on Communication Networks and Power Systems (WCNPS) | 978-1-6654-7456-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/WCNPS56355.2022.9969715

***Abstract*—DNS tunneling uses DNS protocol features to establish command and control channels thus being possibly exploited as a malicious tool for data exfiltration. DNS tunneling security threats affect cross-platform systems within local and cloud computing resources. This article proposes an effective DNS tunnel detection methodology integrating cloud-based resources. The proposed detection methods compose an unsupervised machine-learning model execution for anomaly identification. The validation uses a collected DNS traffic dataset and shows the practical approach for C2, data exfiltration, and heartbeat tunnel test situations, as high levels of anomaly detection are obtained even for those lightweight data during the transfer process. This study has an operational approach and could be adapted to compose security control systems for organizations.**

***Keywords*—DNS Tunneling, Anomaly detection, Cybersecurity, Cloud computing.**

I. Introduction

In recent years, there has been an increase in the number of malware-based attacks that work infecting machines to establish communication channels with remotes servers in order to transfer sensitive and confidential information from organizations. Data exfiltration causes financial losses and affects the reliability of institutions, representing a relevant challenge for cybersecurity. One of the most used method for data leakage is DNS tunneling, using the Domain Name System protocol (DNS) to carry encapsulated information by establishing command and control (C2) channels for remote controls and data manipulation [1].

The DNS tunneling as a cyber attack represents a current concern, mainly due to the efficiency of incidents in the invasion of machines and the inherent difficulty in detection. DNS is a system for translating domains into IP addresses, widely used on Internet, having hierarchical and recursive attributes in communications between authoritative and recursive servers [2]. Due to the essential purpose of the protocol, the DNS UDP/53 port is released from security blocks and, improperly monitored by network security tools.

978-1-6654-7456-6/22/$31.00 ©2022 IEEE

Many studies have been developed to detect DNS tunneling in the last decade, including data analysis and machine learning (ML) to train model algorithms and create inferences. [3] addressed a survey of various combinations of detection techniques tunneling between 2006 and 2020, classifying parameters coming from DNS packets and flows, as well as differentiating the analysis methods between rules, signatures, and based on ML. Other studies ([4] and [1]) focus on the efficient combination of feature selection, DNS tunneling behavior identifiers, and algorithms that classify incidents with the highest accuracy.

ML techniques have been contributing to cybersecurity systems evolution, training models to define patterns, using a sort of classification algorithms, and identifying anomalies. However, organizations have faced practical and operational challenges by the diversity of sources in hybrid networks, managing large volumes of datasets, and applying ML models isolated from the rest of the architecture [5]. Even so, training a model in a different environment to the one deployed would cause a negative performance on real-network productions, despite many research studies [6].

Based on the above considerations, as an operational approach, the detection of DNS tunneled traffic needs to be cross-platform, combining local network resources (on-premise) to ensure dynamically, independently, and scalable security services. In this sense, the objective of this study is a DNS tunneling detection model with a functional proposal. Therefore, we propose an efficient modular architecture, with high levels of accuracy, composed of DNS collections processes from cloud services, flexible dataset storage, features selection based on correlates studies, and unsupervised ML algorithm to identify anomalies.

II. Background

A. Terminology

Table I defines the concepts of the AWS cloud computing platform used to host services and instances for the test topology of this study. The terminologies listed aim to assist in understanding resources, services, programs, and architectural elements to assemble a framework for testing DNS tunneling tools.

Table I: AWS Terminology [7]

Amazon Web Service (AWS)	Cloud computing platform. Flexible resource capacity scaling, agile and on demand.
Elastic Compute Cloud (EC2)	EC2 instances are scalable, on-demand computing resources with a high availability proposition.
Virtual Private Cloud (VPC)	Virtual private network service, logically isolated, that includes IP addressing scopes, creating subnets, and configuring route tables and network gateways.
Route 53	Domain Name System (DNS) service, with three main functions: domain registration, DNS routing and health checking.
Route 53 Resolver	Recursive DNS service that sends requests to authoritative domain servers. The Route 53 resolver is the first to respond to queries for a given VPC.
Simple Storage Service (Bucket S3)	Cloud storage object service. Storage resources management with availability, access controls, backup, and scalability.

B. DNS tunneling attack

DNS tunneling represents a technique that encapsulates data from other protocols, such as SSH or FTP, in DNS requests. Data is encoded and infiltrated into DNS packets for transmission in an established channel between a client (affected machine) and a remote server (attacker). This security threat can exfiltrate information, obtain unauthorized client data, infiltrate malicious code and, transmit remote commands for channel verification and hacked machine inspection.

The DNS tunneling mechanism has a client/server architecture (Figure 1). The attacker holds the malicious domain (*tunnel.abc*), receiving requests by DNS packets, so in a recursive and hierarchical way, the requests from the attacked machine can reach the server (*ex.tunnel.abc*) [8]. The remote server machine process received data using servers scripts modules to successfully decoded by DNS tunnels.

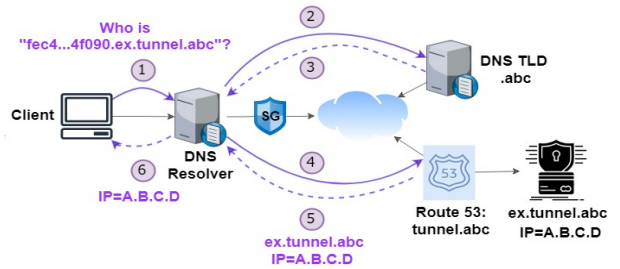


Figure 1: DNS tunneling attack

As a practical description, the main DNS tunneling tools encode information usually in Base128/64/32 or Hexadecimal formats and concatenated data to subdomain queries, must meeting the format size limits imposed by DNS protocol rules [9]. Thereby, a subdomain carrying encoded information for a typical DNS tunneling traffic would have the following format: *ksfiulufpktoxzydegngdsczwsqutee.ex.tunnel.abc*.

Those tools can still use resource records (RRs) fields to insert reported data. The use of some RRs types, such as 'TXT' and 'NULL', increases the transmission bandwidth but does not represent common queries that mainly use 'A', 'AAAA', and 'CNAME' types, effortlessly generating safety alarms. To deviate security controls, DNS tunneling tools may split data into more queries, pause requests, and use a combination of remote servers or prefixes domains with well-known websites.

III. Methodology

This study proposes a modular architecture for malicious DNS tunneling detection, Figure 2. The collection process include cloud-based data from AWS services as well as flexible integrating with third-party cloud platforms, or on-premise environments (dotted-line).

The plaintext nature of DNS traffic makes the detection of tunneling-related activities possible by traffic flow analysis and packet inspection [10], forming two analysis layers for differences response times and future cryptography approach. Processing packets has real-time responses, but reduce the efficiency in detecting anomalies that depend on subsequent packets samples. DNS flow features provide metadata, inferring the behavioral and statistical patterns and can even be used to detect encrypted traffic, although adding overhead due to final flow delay.

Features were splited into dimensions contrasting data with specific influence fields, creating the concept of detectors. In this research, the main dimensions used as influence fields were TLD (Top Level Domain), eTLD+1 (effective Top Level Domain plus one) and IP address resources. Those detectors perspectives aims to drill down into effective tunneling behavior. Subsequently, the Population ML model was chosen as an unsupervised anomaly detection algorithm for issues identification in a IT system monitoring by anomalousness behaviors and DNS tunneling events scoring [11].

For final process, we provided several benign requests queries and DNS tunneling testings attacks, with a combination of tools (Iodine [12], Dnscat2 [13], and Flighsim [14]) and offensive methods likewise command and control (C2), transfer files and heartbeat verification for channel status, to analyze the effectiveness detection levels methodology. All tests results refeeding the model in a continuous unsupervised learning process, becoming more accurately as more data analyzed performed in a time series window.

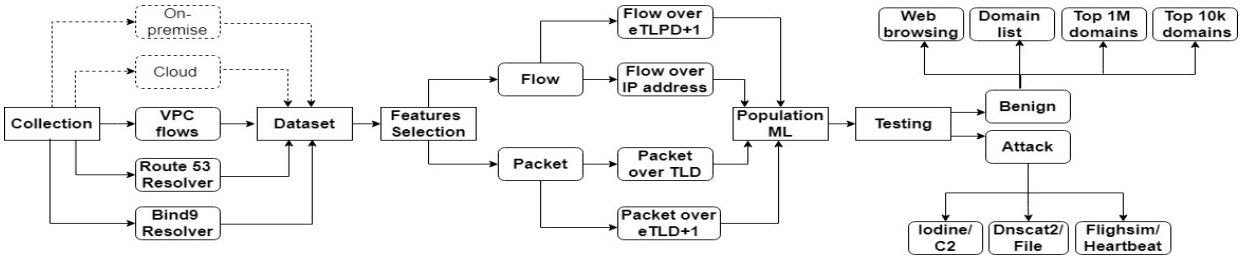


Figure 2: Overall Methodology [2]

A. Architecture

The solution proposed was deployed on AWS, in the IaaS (Infrastructure as a Service) modality, and includes DNS queries and network flows collection by ELK agents (beats) to indexing fields for a dataset construction. Beats delivery indexes mapping and flexibility configuration to improve data categorization. The resulting indexes display in Kibana after sorting, identifying null values, cleaning data and creating new indexes from the combination of raw data. For adjusting real-time analysis, beats were configured to establish a frequent and sequential data pool on endpoint resources to data-feeding the ELK server.

Bind9 represents an endpoint for DNS queries. The Packetbeat module [15] collects DNS packets and flow records for an entire DNS event. Virtual Private Cloud (VPC) and Route 53 Resolver both native AWS services either deployed as endpoints. A VPC flow log service collects network logs, in a 5 minutes period, and send them to an S3 bucket for storage. Likewise, a query logging service from Route 53 Resolver sends queries to another S3 bucket. The Filebeat module [16] was responsible for checking newly registered logs on S3 and transfer to the ELK server.

B. Topology

For test purposes, we set up a private AWS network with internal IP addressing scope segmented into two subnets: one for client machines (EC2 Ubuntu 20.0 and Windows 10 Desktop instances) and another for Bind9 and ELK servers. A testing Kalilinux attacker machine was configured in a different availability zone or external network, receiving DNS tunneling tools modules and listening to requests on UDP/53 port.

EC2 instances have a native domain resolution service on AWS, Route 53 Resolver, but each machine used Bind9 server primarily for name resolution and thus concentrate DNS requests. The choice of an extra endpoint for DNS queries aims to approach different types of logs for indexing, in a complementary way, demonstrating a flexible collection process.

Route53 received the *lsbb.link* domain register and the subdomain *t1ns.lsbb.link* pointing to the remote KaliLinux server. Therefore, this process performs to achieve the attacking server from anywhere on the Internet by recursive access to malicious domains.

C. DNS Features

The main features affected by DNS tunneling processes represent the basic behaviors to construct variables and dimensions for a systematic analysis. The combination of statistical metrics and indexes forms a feature selection process for this proposed anomaly system detection, summarized in Subsection III-D.

Number of eTLD+1 requests: to transmit a series of data tunneling in DNS queries, there is an abnormal increase in the amount of the requests for the same eTLD+1 (Effective top-level Domain plus one), which is the TLD plus one layer in the subdomain.

DNS packets Size in bytes: UDP DNS packets increase when carrying data into subdomains and RR fields. Thus, network byte transfer rates in a DNS event will perform anomalous values.

Resource Records Types: RRs types as A (IPV4), AAAA (IPV6) and PTR (*reverse lookup pointers*) are the most common representing 99,4% of standard requests [17]. Tunneling tools tend to switch or modify the types of RRs in DNS packets (CNAME, TXT, MX, etc.) to increase data bandwidth.

Amount of data transmitted: during a DNS tunneling event, there is an increase in event duration and total data received and sent successfully to the same pair of resource machines.

Time-to-live TTL: tunneled DNS traffic must have the lowest possible TTL so that requests to the malicious domain are not cached on the local resolver server, forcing high cache miss rates [1].

D. Detection Methods

The Population ML model detects unusual activities according to the previous behavior of the population under analysis [18]. The anomaly detection model uses time series and analyzes data in increments of time. Our method approach uses unsupervised learning without any previous assistance for model training. However, the accuracy depends on continuously updating based upon new information [11].

The anomaly detection algorithm splits fields over dimensions and uses probability distribution, such as

Table II: DNS Flow over IP Address and over eTLD+1

DNS flow over destination IP		DNS flow over eTLD+1	
Feature	Detector	Feature	Detector
F1	high_count(destination.address)	F9	max(event.duration)
F2	high_mean(network.bytes)	F10	max (bytes_in)
F3	high_mean(network.packets)	F11	max (bytes_out)
F4	high_mean(source.bytes)		
F5	distinct_count(destination.port)		
F6	distinct_count(related.ip)		
F7	distinct_count(source.address)		
F8	distinct_count(source.port)		

Table III: DNS Packets over eTLD+1 and over TLD

DNS Packets over eTLD+1		DNS Packets over TLD	
Feature	Detector	Feature	Detector
F12	high_count	F21	high_count
F13	distinct_count (dns.question.name)	F22	distinct_count (dns.question.name)
F14	distinct_count (dns.question.subdomain)	F23	distinct_count (dns.question.subdomain)
F15	distinct_count (dns.id)	F24	distinct_count (dns.question.type)
F16	high_mean (dns.answers_count)	F25	distinct_count (dns.id)
F17	low_mean (dns.answers.ttl)	F26	high_mean (dns.answers_count)
F18	distinct_count(dns.answers.name)	F27	high_mean (dns.answers.data)
F19	distinct_count(dns.answers.type)	F28	low_mean (dns.answers.ttl)
F20	high_mean (dns.opt.udp_size)	F29	distinct_count(dns.answers.name)
		F30	distinct_count(dns.answers.type)
		F31	high_mean (dns.opt.udp_size)

Poisson, Gaussian, log-normal, or mixtures of models, to define a behavioral baseline that suits each variable. The analysis is based on the language of hypothesis testing, which describes most of the observations as the null hypothesis. The *p-value* of a test statistic is used for rejecting the null hypothesis and for anomaly detection, identifying a point as an outlier [11].

The scoring of unusual events will be empirically answered with *p-value* number between 0 (no possibility) and 1 (absolute certainty) [19]. The lower *p-value* probability is, farthest from the normal data distribution, and alerts are generated by a normalized score in the range of 0 to 100 (critical above 75).

The features identified in Subsection III-C were combined with counters of high, maximum, mean, and lower values to isolate anomalous situations and create detection methods or jobs. Table II defines the DNS Flow jobs over IP address and over eTLD+1 and Table III lists the DNS Packet jobs within detectors splited over eTLD+1 and over TLD.

IV. Evaluation

A. Dataset

The dataset will consist of normal DNS requests and those generated by DNS tunneling tools in a window of two weeks. For benign data, DNS queries were made to legitimate websites, as well as random web browsing such as email, video streaming, geographic location, news, etc. In addition, sequential query scripts were run for the 10,000 most consulted domains [20] and 1,000,000 top access domain on Internet [21].

DNS tunneling tools were tested: Iodine [12] for C2 tests and Dnscat2 [13] for file transfer, both using the same malicious domain and server attacker *lsbb.link*. Iodine is commonly used for web browsing bypassing captive portals for unauthenticated access on the UDP/53 port. Dnscat2 encapsulates SSH or FTP traffic through DNS queries more efficiently and is more suitable for transferring files.

Flightsim is a lightweight utility used to generate malicious network traffic and help security teams to evaluate perimeter controls [14]. Flightsim simulated DNS tunneling to the *alphasoc.xyz* malicious domain as a heartbeat test and validated our proposed detection model with an extra and unknown tunnel structure.

B. Experimental Results

Flow detection methods: For Iodine, DNS events were considered anomalous and received high scores of 92 and 94 at the two events tested on the dataset due to majority increases in the number of data transferred from the same source and to destination IP, high count of UDP open ports, as well as DNS flows on the network by counting identifier indexes. Such behavior indicates a streaming, continuous transfer on port UDP/53.

For Dnscat2, the same Iodine influencing features prevailed. However, the data transfers on tests increase anomaly by the amount of bytes in transit, both input and output, so those events were even more decisive for identifying the malicious threat (score 96). The flow perspective does not perform for Flightsim utility despite two short shots running tested (about 50 queries for each shot). Minors query counts in DNS flows are silence and represent a normal behavior.

Figure 3a shows the maximum values for received bytes during Dnscat2 tunneling as a sample of an anomaly on the timeline. Another feature, Figure 3b was the anomalies events duration in a DNS tunneling

flow. In both cases, the shadows in gray represent normal points data and the *p-values* probability are very low, about 5.56e-36 for bytes received and 4.07e-9 for events duration, demonstrating an abnormal deviation.

Packet detection methods: Even Iodine tests encapsulated and transmitted only C2 commands (light data), the number of requests per malicious eTLD+1 is relatively high, resulting in an efficient detection (scores 99 and 96 for the two Iodine events).

Dnscat2 performed the highest detection rate (score 99) during the transfer of a lightweight test file *passwords.txt* (100KB), showing that the package features selected in this study were the most effective when used for tools based on managing own subdomains.

The Flightsim, with the purpose of generating a reduced number of queries for a different domain, processing heartbeat requests to check channel status, received a critical score, showing that even with only 1 shot (score 92) and 2 shots (score 93), the model proposed identified efficiently the anomaly, only taking into account packet features in a real-time system.

In Figure 4a, higher number of data inside request answers represents an ordinary technique by DNS Tunneling tools. For each data chunk into queries to transfer information, more subdomains by the same TLD are generated. Figure 4b is an adequate example of a lower feature where the anomaly characterization is by the count of lower values for TTL on DNS packets to identify the need for expired time domain and cache miss force behavior of tunneling method.

C. Considerations

The overall detection methods lead to composing a higher detection rate with all features identified. Figure 5 compares the accuracy scores for anomaly detection by DNS Tunneling tools used in this study.

False positives domains such as *amazonaws.com*, *cloudflare.com*, *akam.net*, are classified as storage content, cloud-hosted services and applications, CDNs (Content Delivery Network), and DNS proxy services. Even though methods scored legitimate events, the malicious domains are still at the top of severity.

Table II features do not deal with any packet fields and can be used to analyze encrypted DNS traffic. Flow features were unsuccessful in detecting tests with reduced and limited numbers of queries, resulting in no outliers identification. However, for packet jobs, the specific data evaluation offers efficient anomaly detection without taking numerous subsequent event windows. The effectiveness detection was important for attacks with lower bandwidths or sparse occurrences in a plain-text DNS context.

A list of reliable domains could still have been applied in this analysis, improving anomaly identification. However, this strategy is a static monitoring manipulation that needs to be constantly updated by security analysts. For those reasons, we decided to evaluate the original samples. It is also important to note that some ransomware-type attacks register domains specifically for the attacks (not matching existing blacklists).

The malicious domain *t1ns.lsbb.link* had the highest abnormal scores in all test events. The tunneled traffic was classified as critical, between 96 and 99, resulting in effectiveness scores mainly for tools using a registered domain with no extra protocol for cryptography. The detection for *alphasoc.xyz* domain, proof of the unsupervised method successfully detected a new DNS tunneling event. the lower scores 92 and 93 were due to reduced counts of queries generated.

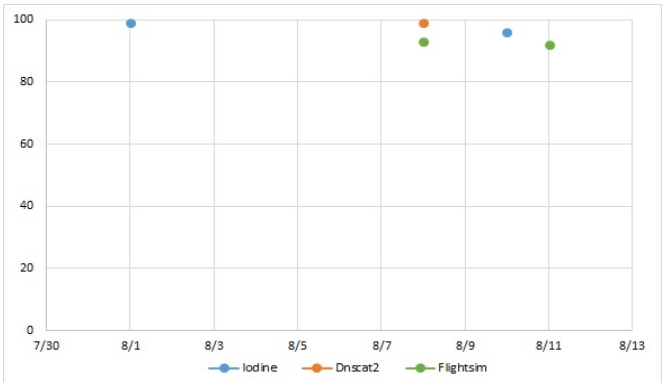


Figure 5: Anomaly Timeline

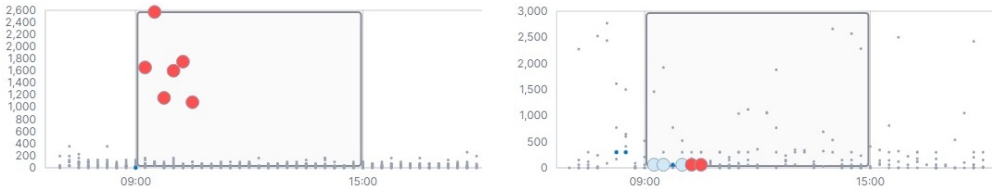
V. Conclusions and Future Work

This study proposed a DNS tunneled traffic detection model that was effective for C2, data exfiltration, and heartbeat tunnel test situations. There were high levels of anomaly detection even for those lightweight data during the transfer process. The malicious events had scored critical levels (ranging from 92 to 99) proving that the anomaly detection model is suitable for this class of threats in terms of accuracy and execution time. The architecture for integrating resources in AWS, collecting, and sending data to the ELK solution has proven to be modular, flexible, and of practical value. The research can be adapted to compose cyber defense solutions for enterprise security teams.

For future works, numerous DNS tunneling tools or malwares can be tested, in addition to training different ML models using the deployed dataset, within new perspectives and inferences to compare. With the proposed architecture and proper flow features layer adapt, it is possible to develop encrypted DNS traffic surveys for DoT (DNS over TLS) and DoH (DNS over HTTPS) protocols. The identification of malicious DNS packets encrypted represents another step of analysis evolution in the detection of tunneled traffic.

(a) DNS flow anomalies received bytes (b) DNS flow anomalies event duration (ns)

Figure 3: Flow features anomalies for Dnscat2



(a) DNS packet outliers answers name (b) DNS packet outliers answers TTL (s)

Figure 4: Packet features anomalies for Dnscat2

Acknowledgement

The authors thankfully acknowledge the support of: CNPq - National Research Council (Grant 312180/2019-5 PQ-2 e 465741/2014-2 INCT Cybersecurity), the Administrative Council for Economic Defense (Grant CADE 08700.000047/2019-14), the General Attorney's Office (Grant AGU 697.935/2019), the National Audit Department of SUS (Grant DENASUS 23106.118410/2020-85), the General Attorney's Office for the National Treasure (Grant PGFN 23106.148934/2019-67), Brazilian Intelligence Agency - ABIN (Grant ABIN 08/2019) and the University of Brasília (Grant FUB/COPEI 7129).

References

[1] N. Ishikura, D. Kondo, V. Vassiliades, I. Iordanov, and H. Tode, "Dns tunneling detection by cache-property-aware features," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1203–1217, 2021.

[2] Y. Khodjaeva and N. Zincir-Heywood, "Network flow entropy for identifying malicious behaviours in dns tunnels," in *The 16th international conference on availability, reliability and security*, 2021, pp. 1–7.

[3] Y. Wang, A. Zhou, S. Liao, R. Zheng, R. Hu, and L. Zhang, "A comprehensive survey on DNS tunnel detection," *Computer Networks*, vol. 197, p. 108322, 2021.

[4] H. Bai, W. Liu, G. Liu, Y. Dai, and S. Huang, "Application behavior identification in DNS tunnels based on spatial-temporal information," *IEEE Access*, vol. 9, pp. 80 639–80 653, 2021.

[5] E. Liberty, Z. Karnin, B. Xiang, L. Rouesnel, B. Coskun, R. Nallapati, J. Delgado, A. Sadoughi, Y. Astashonok, P. Das *et al.*, "Elastic machine learning algorithms in amazon sagemaker," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 731–737.

[6] T. A. Nguyen and M. Park, "DoH tunneling detection system for enterprise network using deep learning technique," *Applied Sciences*, vol. 12, no. 5, p. 2416, 2022.

[7] "Amazon elastic compute cloud documentation," https://docs.aws.amazon.com/, [Online; accessed: 15.10.2022].

[8] D. Tatang, F. Quinkert, N. Dolecki, and T. Holz, "A study of newly observed hostnames and DNS tunneling in the wild," *arXiv preprint arXiv:1902.08454*, 2019.

[9] P. V. Mockapetris, "RFC1035: Domain names-implementation and specification," 1987.

[10] M. Lyu, H. H. Gharakheili, and V. Sivaraman, "A survey on dns encryption: Current development, malware misuse, and inference techniques," *ACM Computing Surveys (CSUR)*, 2022.

[11] T. J. Veasey and S. J. Dodson, "Anomaly detection in application performance monitoring data," *International Journal of Machine Learning and Computing*, vol. 4, no. 2, p. 120, 2014.

[12] E. Ekman and B. Andersson, "Iodine," https://code.kryo.se/iodine/, 2016, [Online; accessed: 05.05.2022].

[13] R. Bowes, "Dnscat2," https://github.com/iagox86/dnscat2, 2015, [Online; accessed: 05.04.2022].

[14] AlphaSOC, "Flightsim utility," https://github.com/alphasoc/flightsim, 2021, [Online; accessed: 08.14.2022].

[15] "Packetbeat references for DNS," https://www.elastic.co/guide/en/beats/packetbeat/current/exported-fields-dns.html, [Online; accessed: 05.08.2022].

[16] "Filebeat for aws s3 input," https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-input-aws-s3.html, [Online; accessed: 05.08.2022].

[17] D. Herrmann, C. Banse, and H. Federrath, "Behavior-based tracking: Exploiting characteristic patterns in DNS traffic," *Computers & Security*, vol. 39, pp. 17–33, 2013.

[18] "Population ML Elastic Stack," https://www.elastic.co/guide/en/machine-learning/current/ml-configuring-populations.html, [Online; accessed: 08.08.2022].

[19] R. Collier and B. Azarmi, *Machine Learning with the Elastic Stack: Expert techniques to integrate machine learning with distributed search and analytics*. Packt Publishing Ltd, 2019.

[20] "Open dns top domains 10k," https://github.com/opendns/public-domain-lists, [Online; accessed: 08.08.2022].

[21] "Open dns top domains 1m," https://github.com/zer0h/top-1000000-domains/blob/master/top-1000000-domains, [Online; accessed: 08.08.2022].

[22] H. Bai, G. Liu, J. Zhai, W. Liu, X. Ji, L. Yang, and Y. Dai, "Refined identification of hybrid traffic in DNS tunnels based on regression analysis," *ETRI Journal*, vol. 43, no. 1, pp. 40–52, 2021.

[23] 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.

[24] S. Mahdavifar, A. Hanafy Salem, P. Victor, A. H. Razavi, M. Garzon, N. Hellberg, and A. H. Lashkari, "Lightweight hybrid detection of data exfiltration using dns based on machine learning," in *2021 the 11th International Conference on Communication and Network Security*, 2021, pp. 80–86.