Network-Based Detection and Prevention System against DNS-Based Attacks

Yasir Faraj Mohammed
University of Arkansas, Fayetteville

Network-based Detection and Prevention System against DNS-based Attacks

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Engineering with a concentration in Computer Science

by

Yasir Faraj Mohammed
AL-Hadbaa University
Bachelor of Science in Computer Science, 2004
Glamorgan Uninversity
Master of Science in Computer System Security, 2009

May 2021
University of Arkansas

This dissertation is approved for recommendation to the Graduate Council

ABSTRACT

Individuals and organizations rely on the Internet as an essential environment for personal or business transactions. However, individuals and organizations have been primary targets for attacks that steal sensitive data. Adversaries can use different approaches to hide their activities inside the compromised network and communicate covertly between the malicious servers and the victims. The domain name system (DNS) protocol is one of these approaches that adversaries use to transfer stolen data outside the organization's network using various forms of DNS tunneling attacks. The main reason for targeting the DNS protocol is because DNS is available in almost every network, ignored, and rarely monitored. In this work, the primary aim is to design a reliable and robust network-based solution as a detection system against DNS-based attacks using various techniques, including visualization, machine learning techniques, and statistical analysis. The network-based solution acts as a DNS proxy server that provides DNS services as well as detection and prevention against DNS-based attacks, which are either embedded in malware or used as stand-alone attacking tools. The detection system works in two modes: real-time and offline modes. The real-time mode relies on the developed Payload Analysis (PA) module. In contrast, the offline mode operates based on two of the contributed modules in this dissertation, including the visualization and Traffic Analysis (TA) modules. We conducted various experiments in order to test and evaluate the detection system against simulated real-world attacks. Overall, the detection system achieved high accuracy of 99.8% with no false-negative rate. To validate the method, we compared the developed detection system against the open-source detection system, *Snort* intrusion detection system (IDS). We evaluated the two detection systems using a confusion matrix, including the recall, false-negatives rate, accuracy, and others. The detection system detects all case scenarios of the attacks while Snort missed 50% of the performed attacks. Based on the results, we can conclude that the detection system is significant and original improvement of the present methods used for detecting and preventing DNS-based attacks.

ACKNOWLEDGMENTS

First and foremost, I am extremely grateful to my supervisor, Dr. Dale R. Thompson, for his invaluable advice, continuous support, and patience during my Ph.D. study. His immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. Finally, I would like to express my gratitude to my parents, my wife, and my children. Without their tremendous understanding and encouragement in the past few years, it would be impossible for me to complete my study.

DEDICATION

To my parents, my wife and my children, Duaa, Rahaf, Amro, Joury.

TABLE OF CONTENTS

1 Introduction . . . 1
1.1 Background . . . 1
1.1.1 DNS Protocol . . . 1
1.1.2 DNS Namespace Hierarchy . . . 2
1.1.3 Data Exchange over DNS . . . 4
1.1.4 DNS Attacks . . . 5
1.1.5 DNS Tunneling Attack . . . 6
1.1.6 Communication Patterns . . . 6
1.1.7 Visualization . . . 7
1.1.8 Machine Learning Techniques . . . 7
1.2 Statement of Problem . . . 8
1.3 Goals of this work . . . 9
1.4 Contributions . . . 10
1.5 Dissertation Organization . . . 11

2 Related Work . . . 12
2.1 Detection of DNS attacks . . . 13
2.2 Detection DNS Tunneling attacks . . . 14
2.3 Detection IDS/IPS systems . . . 15
2.4 Detection using Machine Learning . . . 17
2.5 Detection using Visualization . . . 18
2.6 DNSSec . . . 19

3 Visualization of DNS Tunneling Attacks . . . 21
3.1 Introduction . . . 21
3.2 Visualization System Design . . . 22
3.3 Graphical patterns for DNS tunneling attacks . . . 23
3.4 Evaluation . . . 26
3.4.1 Experimental setup . . . 26
3.4.2 Results . . . 27
3.4.3 Issues . . . 29
3.5 Conclusion . . . 30

4 DNS Tunneling Attack Detection using Machine Learning . . . 31
4.1 Methodology . . . 31
4.1.1 System Design . . . 31
4.1.2 The Detection System . . . 33
4.2 Results and Discussion . . . 35
4.2.1 Evaluation . . . 35
4.2.2 Experimental Setup . . . 35
4.2.3 The Dataset . . . 38
4.2.4 Evaluation Metrics . . . 40
4.2.5 Results . . . 43

5 System Methodology . . . 46
5.1 Background . . . 46
5.2 DNS Proxy . . . 46
5.2.1 DNS Proxy as a Detection System . . . 47
5.3 System Design . . . 48
5.3.1 Parsing DNS Data . . . 48
5.3.2 Feature Extraction . . . 49
5.3.3 The Payload Analysis Module . . . 50
5.3.4 The Traffic Analysis Module . . . 53
5.3.5 System Features . . . 56

6 Results and Discussion . . . 58
6.1 Evaluation . . . 59
6.1.1 Experimental Setup . . . 59
6.1.2 The Collected DNS Data . . . 60
6.2 Threshold Value . . . 63
6.3 Results . . . 65
6.3.1 Results of Experiment 1 . . . 67
6.3.2 Results of Experiment 2 . . . 69
6.4 Results . . . 69
6.4.1 Snort . . . 69
6.4.2 Snort Setup . . . 69
6.4.3 Snort Results . . . 71
6.4.4 DNS Proxy vs Snort . . . 71

7 Conclusion and Future Work . . . 75
7.1 Conclusion . . . 75
7.2 Discussion and Limitations . . . 77
7.3 Future Work . . . 78

Bibliography . . . 79

LIST OF FIGURES

Figure 1.1: The Domain Name System Hierarchy 2
Figure 1.2: How DNS works 3
Figure 1.3: Data exchange over the DNS protocol 5
Figure 1.4: The DNS tunneling attack 6

Figure 3.1: System Architecture Design 23
Figure 3.2: Graphical Patterns for Traditional DNS Tunneling Attack. 25
Figure 3.3: Graphical Patterns for Heyoka DNS Tunneling Attack. 26
Figure 3.4: Top Five Domain Name for an Entire Week without the Attack. 27
Figure 3.5: Top Five Domain Names during the Attack over a Two-hour Period. . . 28
Figure 3.6: Visualizer Module Shows only the Attack. 29
Figure 3.7: Visualize the Entire DNS Captured Data. 29

Figure 4.1: The DNS Tunneling Attack Patterns 32
Figure 4.2: DNS Tunneling Tools 36
Figure 4.3: Top 25 Domain Names including DNS Tunneling Attack Tools 37
Figure 4.4: The Normal and the Malicious DNS requests in the dataset 39
Figure 4.5: The Total Number of the Normal and the Malicious DNS requests in the dataset 40
Figure 4.6: The Total Number of the TTL and Data Length in the Normal and the Malicious DNS requests in the dataset 41
Figure 4.7: The Confusion Matrix 41

Figure 5.1: DNS Proxy 47
Figure 5.2: The Architecture Design of DNS Proxy Detection System. 49
Figure 5.3: The Real-time Payload Analysis Module. 53
Figure 5.4: The Overview of Traffic Analysis Module. 54
Figure 5.5: Example of Parsed DNS Data 54

Figure 6.1: An Overview of the DNS proxy Setup. 60
Figure 6.2: Internet Data Transfer during the Four Weeks. 61
Figure 6.3: DNS Queries over Monthly Time Period. 62
Figure 6.4: Statistics of Queries Types and Answered by. 63
Figure 6.5: Environmental Setup of Snort. 70
Figure 6.6: Snort TXT Signature type for DNS Tunnelling. 71

LIST OF TABLES

Table 1.1: Well-known malware that uses DNS protocol for their communications . . 9

Table 4.1: DNS Tunneling Visualization Signatures 33
Table 4.2: Frequency of Top 10 Domain Names 38
Table 4.3: The confusion matrix values for the machine learning classifiers. 44
Table 4.4: Compared Results Among Machine Learning Classifiers 45

Table 5.1: DNS Tunneling Visualization Signatures 55

Table 6.1: Frequency of Top 10 Domain Names 62
Table 6.2: Exfiltrating Various Binary files Over the DNS 65
Table 6.3: Tools to Simulate DNS-Based Attacks 66
Table 6.4: Frequency of Top 10 Blocked Domain Names 67
Table 6.5: Top 5 Domain Names of False Positives 68
Table 6.6: Compared Results between DNS Proxy and Snort 71
Table 6.7: Comparison of Three Experiments Results 72
Table 6.8: The Major Difference between the Detection System and Snort IDS . . . 73
Table 6.9: Features Used in Detection DNS Tunneling and DNS Exfiltration 73

1 Introduction

The Internet has become a significant part of people's daily life and an essential method for communications today. Individuals and companies rely on the Internet and use it as a primary tool for doing their personal or private business. Thus, personal computers and computer networks have been a primary target of data theft, malware software, and other types of attacks such as accessing, changing, or destroying sensitive information that interrupt normal business processes. This requires cybersecurity practices to protect systems, networks, and applications from various cyber attacks. Cybersecurity provides multiple layers of protection deployed among different components of network infrastructures to block the Internet-based attacks and includes firewalls and intrusion detection systems (IDSs). However, sometimes adversaries can bypass these defensive mechanisms by using different techniques. For example, the Domain Name System (DNS) is the critical backbone of the Internet providing the mapping of human-readable names to Internet Protocol (IP) addresses used for communication. Therefore, restricting the DNS protocol in a firewall or security appliances is challenging, and it may raise usability issues since almost every communication session on the Internet relies on the DNS protocol. Thus, the DNS protocol is one of the most laborious protocols that must be maintained for reliable communication inside a network environment.

1.1 Background

In this section, background information for this work is presented. The background includes a description of the DNS protocol, its usage, and DNS attacks. In addition, data exfiltration techniques and DNS tunneling attacks are introduced.

1.1.1 DNS Protocol

Network infrastructures are often the primary target of adversaries. If any of these services go down or are not working probably, such as DNS, it can cause severe damage to an organization's business. DNS plays an essential role in the Internet architecture and most of the Internet and internal network communications rely on the DNS protocol. It is defined as a client-server protocol for exchanging names and IP addresses. The DNS protocol is a distributed hierarchical database which stores information about a group of registered computers on the Internet. There are different types of information that the DNS protocol can store such as IP addresses including IP version 4 (IPv4) and IP version 6 (IPv6), hostnames, and mail routing information.

1.1.2 DNS Namespace Hierarchy

The DNS has a hierarchical tree structure called the DNS namespace. Each dot in a domain name indicates the separation between levels in the tree structure. The top layer of the DNS tree structure represents the root level which starts with a dot. Under the root level, there are top-level domains (TLD) which are children domains of the root such as .com, .net, .edu and others. Next, TLD also has children domains that reference the second level of domains or authoritative domain name servers. Finally, the fully qualified domain name (FQDN) locates the hostnames or subdomains within the DNS hierarchy [1]. For example, the process of looking up the following domain name *csce.uark.edu.* is always started with the dot which is the root server in the DNS tree structure. Then, the root server(s) forwards the query to the correspondence TLD which in this case is *.edu*. Next, the TLD sends the query to the secondary level domains. As soon the query is received by the authoritative domain name server, it looks up the subdomain equivalent and returns its IP address. Figure 1.1 illustrates the DNS hierarchy.

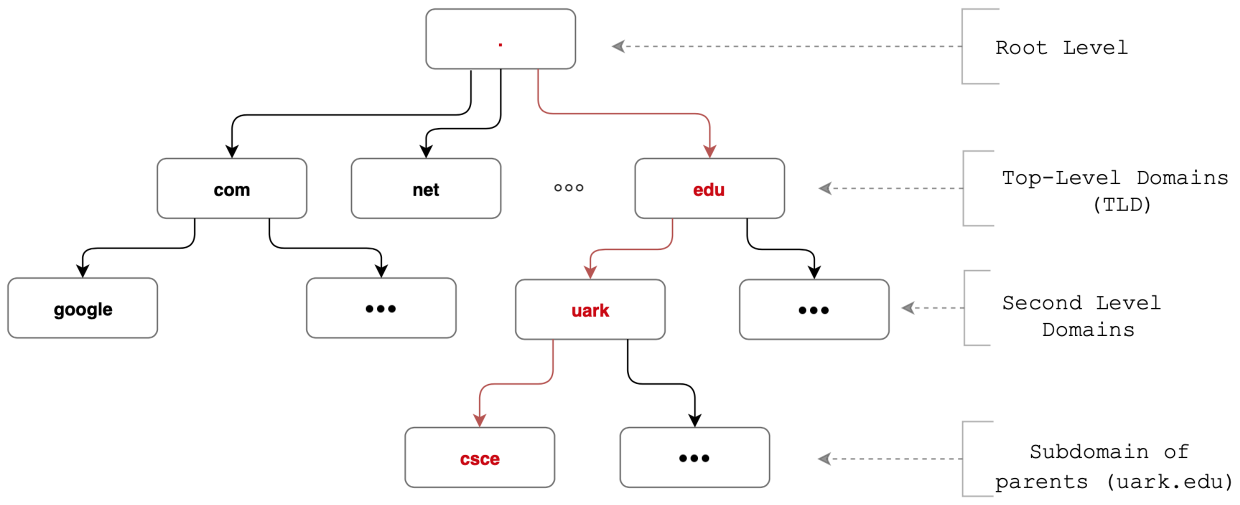


Figure 1.1: The Domain Name System Hierarchy

DNS translates and locates easy-to-remember domain names into corresponding IP addresses and vice-versa. As seen in Figure 1.2, it is assumed that a user types a domain name *example.org* in a browser on a computer. First, the browser attempts to resolve the IP address of the domain name by checking the local cache of the computer. If there is no information related to that domain name and depending on the network configuration, it often forwards a query as a question seeking to match the domain name with its corresponding IP address to the local DNS server or the Internet Service Provider's (ISP) DNS server. Next, if the information is still not found, the request is forwarded to root nameservers. The root servers are located all over the world that point to the appropriate downstream name servers. Then, it gets forwarded to the relevant TLD server which stores the address information for second level domains *example.org* within the top-level domain *.org*. As soon as the information is gathered from authoritative DNS servers, it resolves the domain name and responds to the initial sender. Finally, the computer communicates with a web server to display the content of the website.

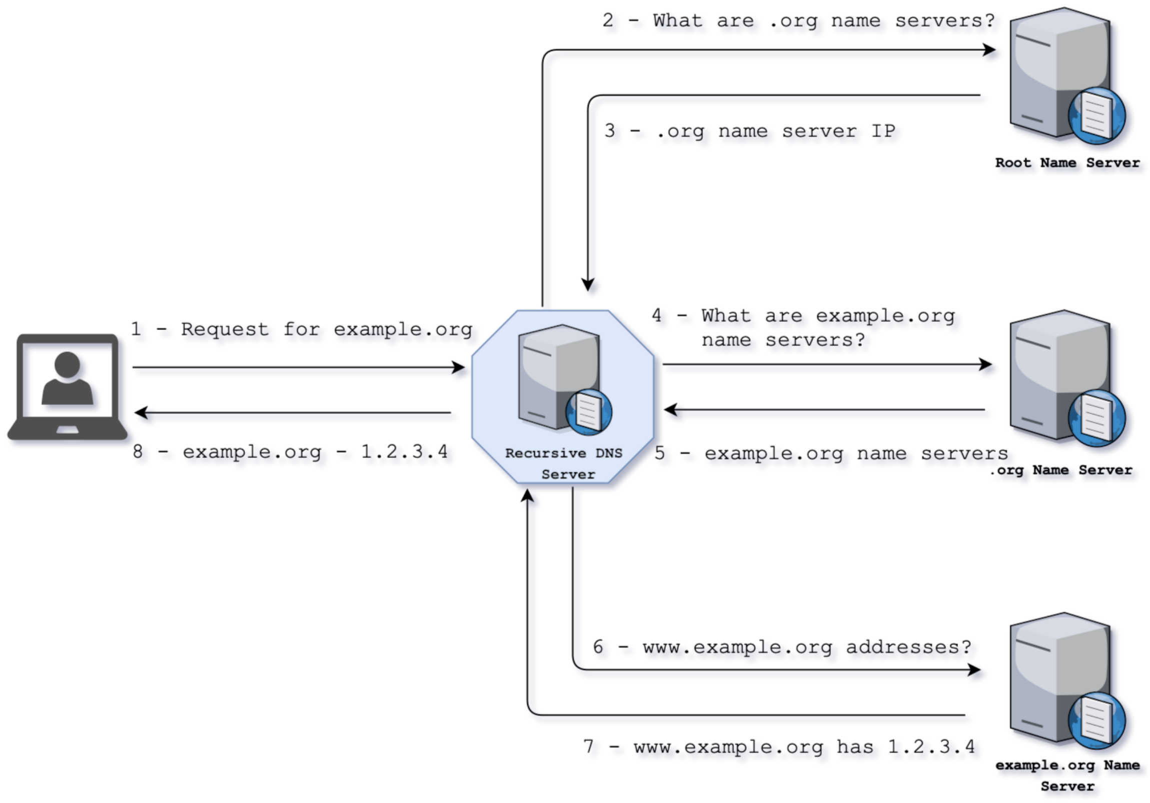
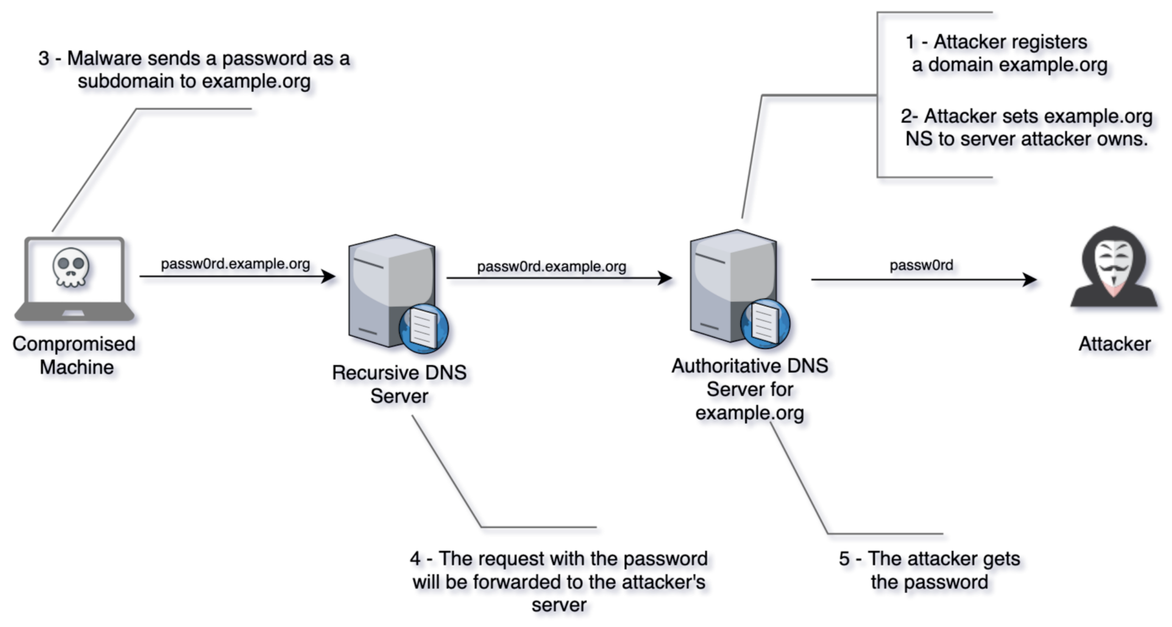

2 - What are .org name servers?
3 - .org name server IP
Root Name Server
1 - Request for example.org
4 - What are example.org name servers?
8 - example.org - 1.2.3.4
Recursive DNS Server
5 - example.org name servers
.org Name Server
6 - www.example.org addresses?
7 - www.example.org has 1.2.3.4
example.org Name Server

Figure 1.2: How DNS works

1.1.3 Data Exchange over DNS

The DNS is a stateless protocol, meant to receive and answer requests from clients. The DNS protocol was not designed for data transfer. It was designed to exchange very short and specific types of information [2]. According to [3], the length of any label of the domain name is limited to between 1 and 63 bytes, and a full domain name is limited to 255 bytes if letters, digits, hyphens, and separators are included. However, the DNS protocol is considered an excellent covert channel from a security perspective. A covert channel is a channel used to transfer information between processes in a secretive manner, bypassing the security policy of a system. Thus, adversaries may abuse and leverage the DNS protocol to transfer data to their authoritative name server. Instead of sending legitimate subdomains, adversaries take advantage of the limitation of the domain length and encode some data in the domain name. For example, if the *passw0rd.example.org* domain is queried, the authoritative name servers of *example.org* receives the *passw0rd* as a string and interpret it as incoming data instead. Also, adversaries may use some other encoding techniques in order to transfer text or binary data to their servers. For example, they can use *Base32, Base64, and Based128* to encode their data. In this case, the query will be as the following: *YWRtaW5AOnBhc3N3MHJkCg.example.org* where *YWRtaW5AOnBhc3N3MHJkCg* is a base64 form of the sensitive text file and this is called a *DNS data exfiltration technique.* Note that if the required data exceeds the 255 bytes, then it has to be split into more than one DNS query in order to be transferred via the DNS protocol. Data exchange over the DNS protocol is shown in Figure 1.3

Although DNS is not intended to exchange data over its protocol, there is a limited number of legitimate data exchange uses over the DNS protocol. Antivirus software companies were leveraging this technique to operate their services to clients. In 2007, Trend-Micro Inc. leveraged the DNS protocol to distribute malicious code signatures for updating antivirus client software as an alternative communication channel. The signatures are exchanged using DNS protocol *TXT* resource records and encoding the data with Base64 [4]. Similarly, McAfee global threat intelligence (GTI) serves its file reputation service through the DNS protocol. It enables clients to post a suspicious file through DNS queries to be scanned. GTI File Reputation technology expands McAfee product security capacities by offering access to an Internet cloud database that contains information on file classification to determine if a file is malicious [5]. In 2009, Devicescape Software Inc. introduced an impressive pub-



3 - Malware sends a password as a subdomain to example.org
1 - Attacker registers a domain example.org
2- Attacker sets example.org NS to server attacker owns.
passw0rd.example.org
passw0rd.example.org
passw0rd
Compromised Machine
Recursive DNS Server
Authoritative DNS Server for example.org
Attacker
4 - The request with the password will be forwarded to the attacker's server
5 - The attacker gets the password

Figure 1.3: Data exchange over the DNS protocol

lic hotspot authentication system for mobile devices that leveraged the DNS protocol for their communications where mobile devices exchange information and media access control (MAC) addresses through the DNS protocol to communicate with a credential server [4]. In addition, the malicious use of data exfiltration over the DNS occurs from other software and malware, which are discussed later in this chapter.

1.1.4 DNS Attacks

The DNS protocol is vulnerable to several attacks such as cache poisoning and DNS hijacking attacks that lead to compromising a user's online account [6]. The DNS amplification attack is a type of Distributed Denial of Service (DDoS) attack based on the DNS protocol that may prevent visitors from reaching the organizations services [7]. There are also other DNS attacks including DNS tunneling, name collisions, leaked queries, and DNS response modification [8]. Most organizations pay more attention to filtering and securing common protocols such as HTTP or FTP from malicious activities instead of the DNS protocol. Since the DNS protocol is not designed for data transfer, monitoring DNS has not received much attention. Over the years, adversaries found different ways to leverage DNS systems for malicious purposes. For example, suppose an organization blocks all outbound traffic except the DNS protocol and DNS activities are not monitored. Adversaries can still transfer stolen or sensitive information through a firewall using DNS. They take advantage of this open avenue and manipulate the use of the DNS service to establish undetected covert channels.

1.1.5 DNS Tunneling Attack

The DNS tunneling attack is a method that encodes and encapsulates the data of other applications or protocols in DNS queries and responses and is shown in Figure 1.4. The original concept of DNS tunneling attack was designed to bypass the captive portals for paid Wi-Fi service, especially at hotels and cafes. However, a data payload can be added to attack internal networks that are used to control remote servers and applications, or to perform DNS data exfiltration which is a technique being used to transfer unauthorized data between two computers through the DNS protocol [9]. To successfully execute this attack, an adversary requires a compromised machine within the internal organization network that has access to the internal DNS server which has external access via the DNS protocol. In this way, the adversary can use the compromised machine to establish a connection to an outside computer using the DNS protocol. Adversaries must also register a domain name and set up a remote server that can run as a DNS authoritative server.

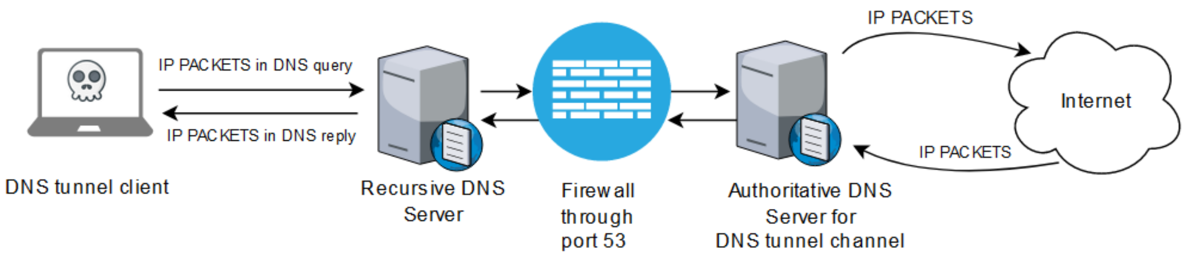


Figure 1.4: The DNS tunneling attack

1.1.6 Communication Patterns

DNS tunneling and data exfiltration communication patterns are significantly different from the normal DNS patterns and from each other. The communication channel of DNS tunneling attack, which can be performed using software such as *Iodine*, *DNScat*, *DNS2TCP*, and others, is more reliable than the DNS data exfiltration techniques. The DNS tunneling attack communication has many keep-alive messages with bi-directional and interactive patterns. Due to the DNS limitation of 255-byte messages, a high volume of DNS queries is required. Therefore, a system administrator could quickly notice the attack if the DNS protocol is monitored. The common usage of DNS tunneling attack is web browsing over the DNS and remote desktop protocols. On the other hand, DNS data exfiltration is less aggressive than DNS tunneling. The communication channel of DNS data exfiltration is built on uni-directional patterns which does not require DNS replies from an authoritative DNS server. DNS data exfiltration is usually performed by malware and command and control (C&C) channels to transfer sensitive data, such as stolen credit cards, outside the network through the firewall. In this technique, adversaries use common DNS records such as A and $AAAA$, which makes the detection more difficult than the others.

1.1.7 Visualization

Visualization is not a new concept for monitoring network traffic. There are several papers that used the parallel coordinates technique to monitor and visualize network traffic [10, 11]. There are various benefits of using a parallel coordinates technique. First, it is scalable. It helps to represent a considerable amount of data in an efficient manner which makes analyzing big data more simple and powerful. Second, there is no limit on the number of values for representing an attack using the parallel coordinates technique that converts multidimensional data into two dimensions. Therefore, using the parallel coordinates technique gives the ability to represent more than three different values within a two-dimensional space. In this way, the relationship between the results can be reviewed quickly showing prominent trends, correlations, and divergence from the raw data. Finally, the visualized data using the parallel coordinates technique does not include any bias for any column within the space [12, 13]. In other words, every visualized feature has the same weight.

1.1.8 Machine Learning Techniques

Machine Learning is a part of Artificial Intelligence (AI) applications, and it is a data analytics technique that provides the capability of systems for automatically learning and improving from experience without human interaction. Machine learning algorithms use computational approaches to *learn* information directly from data without relying on a predetermined equation as a model.

Machine learning consists of three primary classifications: supervised learning, unsupervised learning, and semi-supervised learning. Supervised learning includes labeling information on training data using classification algorithms to predict new data labels. On the other hand, unsupervised machine learning uses unlabeled information and builds a classifier based on learning and information characteristics analysis. It tends to automatically cluster data. Finally, semi-supervised learning is a classifier using both labeled and unlabeled data to train the algorithm [14]. Nowadays, machine learning is actively being used for various topics, including detection systems against malicious activity. In Chapter 2, we list several proposed systems that use machine learning for detecting abnormal activities including DNS tunneling, data exfiltration, and botnets that use DNS protocol for communications.

1.2 Statement of Problem

Detecting and preventing DNS-based attacks is a challenging task. Adversaries use DNS-based attacks to transfer sensitive data, such as stolen private organization data, or establish covert communication channels to hide their malicious activities. DNS-based attacks happen often and can result in a loss of revenue. The 2020 Global DNS Threat Report revealed that 79% of organizations had experienced DNS-based attacks, including DNS phishing, DNS-based malware, and DNS tunneling attacks [15]. Moreover, DNS-based attacks were the main cause for 82% of application downtime in 2020, which caused approximately more than 1 million dollars of damage to these organizations. This work could mitigate these attacks.

Adversaries can take advantage of DNS protocol and establish a covert channel since it is allowed in almost every network, ignored, or rarely monitored. Many research groups have proposed solutions for monitoring and detecting against DNS attacks without focusing on the DNS tunneling attack. The data exfiltration over the DNS protocol and DNS tunneling attack have not received much attention since it was proposed in 2008. As mentioned in Farnham and Atlasis (2013), using the DNS tunneling attack, people are able to bypass most of the firewall rules, restrictions and captive portals for paid Wi-Fi service and also can transfer stolen or sensitive data to outside of the organization's network [9]. Nowadays, most of the modern firewalls and IDS appliances do not block a DNS tunneling attack or raise the alarm by default. For that reason, the DNS tunneling attack is attractive to adversaries or people who design malicious applications.

The data exfiltration over the DNS protocol and DNS tunneling attacks are being used in many real-world scenarios such as malware, botnets, trojans, and others. According to [2], there are various types of well-known malware software that targets victims using the DNS protocol for creating covert communication channels. Table 1.1 shows more information about malware software from 2011 to 2017 which were explicitly designed to avoid detection systems.

Year	**Malware**	**Targets**
2011	Morto	RDP/RAT
2011	FeederBot	Botnet
2014	Plugx	RDP/RAT
2014	FrameworkROS	POS
2015	Wekby	Targeted
2015	BernhardPOS	POS
2015	JAKU	Botnet
2016	MULTIGRAIN	POS
2017	DNSMessenger	Targeted

Table 1.1: Well-known malware that uses DNS protocol for their communications

1.3 Goals of this work

The objectives of this work are listed below:

1. The DNS protocol is vulnerable to various categories of attacks, including DNS protocol attacks, DNS server attacks, and DNS abuse. In this work, the main goal is to detect and prevent DNS-based attacks based on the DNS abuse category. DNS-based attacks include DNS tunneling, DNS exfiltration, and command-and-control communication channels.

2. Design and implement a real-time detection mode that identifies the DNS-based attacks

based on the Payload Analysis module. This module parses, extracts, and analysis captured fully qualified domain name (FQDN) queries from clients to determine whether malicious or not.

3. Design and implement an offline detection mode to distinguish abnormal activity within the DNS traffic. This module works based on two modules: the visualization and the machine learning-based detection modules. The offline detection mode feeds the system with DNS traffic based on a specific window period, every x minutes, to determine whether the DNS traffic contains an attack.

4. Implement and design a network-based solution, DNS proxy server, that combines the two detection modes in order to detect and prevent DNS-based attacks. The network-based solution in this work aims to provide DNS services as well as the detection and prevention in high accuracy, scalability, and efficient manner.

5. Compare and evaluate a set of performance metrics with different detection systems or previous research work.

1.4 Contributions

The contributions of this work include a visualization system, a machine learning-based detection system, and a comprehensive detection system that acts as a DNS proxy server for identifying DNS-based attacks. The contributions of this work include: (1) A fast and scalable visualization system for recognizing DNS tunneling attacks using the parallel coordinates technique. This module is able to distinguish between normal and malicious DNS traffic. (2) Identifying four different graphical patterns of real-world DNS tunneling attacks by analyzing the DNS tunneling traffic based on different scenarios. (3) Designing a module that employs a machine learning classification to predict DNS-based attacks that trained and tested by comparing the graphical representation signatures of the measured network traffic features with the previously developed DNS tunneling attack graphical patterns. (4) Designing a real-time detection module based on the payload analysis to detect DNS-based attacks in real-time. This module works based on malicious DNS-based characteristics. This module achieved high accuracy and no false-negatives values with low false-positives rate. (5) Implementing detection and prevention mechanisms against DNS-based attacks by using the concept of DNS proxy that includes and combines the presented modules in this work.

(6) Designing a system to detect DNS tunneling attacks that achieves a high accuracy of 99.8% and no false negatives with a low false-positives rate of less than 0.02%. The system performs better than the commonly used *Snort* Intrusion Detection System (IDS), and the detection system in this work to determine which system obtained better performance and results against various DNS-based attacks.

1.5 Dissertation Organization

The rest of this dissertation is organized as follows. Chapter 2 discusses the related work to the visualization and detection systems of DNS tunneling attacks. Chapter 3 describes the methodology of the visualization and detection module and the evaluation of the visualization and detection module, which is discussed. The machine learning-based detection module is presented in Chapter 4 and its evaluation and results. Chapter 5 includes the DNS proxy detection system, which combines all detection modules as well as the real-time analyzer module. The evaluation and the results of the DNS proxy detection system are discussed in Chapter 6. Finally, the conclusions and planned future work are presented in Chapter 7.

2 Related Work

Sensitive data theft is now one of the most severe risks for organizations and service providers involving man-in-the-middle attacks [16] or malware that leaks data overt a cover channel [17]. Although many organizations have deployed traditional security systems such as firewalls, intrusion detection systems, and proxies, there are still high-profile data breaches that occur. Marriott International announced on November 2018 that it had been the victim of a data breach in which the information for approximately 500 million clients had been stolen [18]. The breach occurred on Starwood brand support systems starting in 2014. The breach was not discovered until September 2018 [18]. In 2016, Yahoo announced that it was the victim of the greatest data breach in history. The attack compromised approximately one billion user accounts, real names, email addresses, birth dates, passwords, and telephone numbers [19].

The SANS data protection survey revealed in 2017 that 12% of business organizations surveyed recorded a security breach and 43% of these breaches involved data exfiltration that was executed over the DNS [20]. DNS is one of the essential elements of the Internet architecture, and the primary purpose of it is to provide a lookup mechanism to resolve domain names to IP addresses and vice-versa. Given that DNS is not intended for arbitrary data exchange via the protocol, individuals and organizations do not pay attention to security issues in DNS which makes malware masters of abusing the DNS to cover their activities within the networks. Previously, botmasters and hackers were using IRC as communication channels for their bots to avoid various network defense borders [21]. However, nowadays, DNS protocol is being used instead for several reasons: DNS is allowed in almost every network, rarely monitored, and network administrators do not pay attention to secure it. In 2017, Cisco Security revealed that 91% of malware used the DNS protocol for communications, and 68% of organizations did not to monitor DNS traffic [22]. According to SonicWall's Internet security report, in the first half of 2019, there was an increase in the number of malware attacks by 4.8 billion [23]. Based on these Internet security reports, the current real-time detection and prevention systems of malware attacks are not sufficiently capable. Thus, there is still a need in building a robust real-time detection and prevention system in order to decrease the risk against such attacks.

2.1 Detection of DNS attacks

Compared to other protocols, the monitoring of the DNS protocol does not get much attention in the literature. However, there is research on monitoring and the analysis of DNS service without focusing on the DNS tunneling attack as done in this work. The authors in [24] gathered DNS responses data including squatter domains, fast flux domains, and domains being abused by a spammer of the University of Auckland and stored the data into an SQL database for analyzing and detecting unusual behavior. Their research focuses on analyzing DNS traffic to detect spam attacks using statistical analysis instead of visualization. They used the Microsoft Strider URL Tracer, which helps users to investigate third-party domains to crawl data. In this way, they detect that the client has already visited a malicious domain name and has already been compromised. In [25], the efficient detection method of suspicious DNS traffic by resolver separation per application program is proposed. Their approach is based on making further investigation easy by a separate DNS resolver based on the application program. The DNS requests that are sent from Internet Explorer or any registered application are forwarded to a regular DNS resolver while all other unregistered applications' requests are forwarded to a more secure DNS resolver that inspects DNS traffic. The authors point out that this approach helps system to filter out and monitor the suspicious DNS traffic from unknown resources. The proposed system is implemented and tested on the Windows operating system, and it requires installing a Perl module on each endpoint computer to forward DNS redirect traffic to a DNS proxy. As a result, the proposed system was able to forward the DNS queries based on application correctly, and it was able to register the client by mapping the DNS queries with the respective application program. However, this work has some limitations. The user must manually register the applications to be well-known in their system. They also define the suspicious DNS traffic as queries that are sent from not registered official applications in the DNS proxy. The proposed method only supports Windows operating system only. It also hard to register an application that uses multiple processes. In addition, attackers or malware creators can bypass this method by using code injection techniques or DLL injection to insert malicious code into any of the well-known processes. This work is evaluated based on forwarding the DNS requests only as they mentioned in future work. They are planning to evaluate their system against real malware. In [26], the authors described and evaluated a proposed solution for detecting DDoS attack amplifiers attacks, which is mainly designed to efficiently protect the local DNS servers against rogue DNS servers. Further experimental results showed that it is efficient and can be easily fit correctly into any network domain. The proposed system in [4] works as passive DNS that monitors all DNS requests, including queries and responses, and detects payload distribution channels that are established within DNS messages using a DNS zone analysis profile module. Their detection module works based on all *TXT* records that have been captured, divided and aggregated based on one day period. In parallel, their system stores a copy of the captured data using the passive DNS channel in a database for future use. Three datasets have been used to evaluate their work: one month of passive DNS traffic, a passive DNS database, and a one-year malware database. Their experimental results show that the proposed system was able to find different malware that leverage DNS for their data transfer. However, the proposed system has several limitations. For example, the author decided to captured only DNS traffic that has *TXT* recoded to detect the payload distribution channels. In [27], the authors presented a comparative performance evaluation of DNS tunneling tools, and they showed that different tools are able to use not only *TXT* but also different types of DNS records in order to perform the DNS tunneling attacks. Botmasters or malware creators can bypass the proposed system by using their name servers or other open DNS resolvers which are not captured by the passive DNS sensors. As they mentioned in the paper, the proposed system could be configured to run in real-time. However, the system captured and aggregated DNS traffic in a time-based window, of one-day. Therefore, by that time, malware can completely transfer sensitive data outside the organization.

2.2 Detection DNS Tunneling attacks

The DNS tunneling attack introduced in this work is a significant concern in cybersecurity. As mention in [28], the attack can be divided into three subcategories where attackers can abuse the DNS protocol: Data exfiltration, DNS tunneling, and command & control. Data exfiltration is the simplest type of the attack where each DNS packet encapsulates information and then reconstructs this information on the attacker's authoritative DNS server. The main difference between data exfiltration and the other subcategories is that the communication channel is uni-directional. This type of attack is difficult to detect because there is no need to reply to the initial request as this attack transfers sensitive data outside the organization's network. Data exfiltration is often referred to as low throughput exfiltration [2]. Next, the DNS tunneling attack is intended to encapsulate other protocols such as HTTP or FTP within the DNS requests. Compared to data exfiltration, this attack requires a high frequency and volume of DNS traffic that produces easily identifiable traffic pattern. The DNS tunneling attack is often referred to as high throughput tunneling [2]. Finally, the command & control, C&C, communication channel provides the attacker the ability to take full control of a victim computer by sending commands and receiving the corresponding result over the DNS protocol. C&C channels provide an attacker with the means to communicate with their malware after it has infected a victim [28]. Many open-source DNS tunneling tools were designed to perform the attack such as *iodine*, *dns2tcp*, *DeNiSe*, *dnscapy*, *heyoka*, *fraud-bridge*, *dnscat*, and *dnscat2*[9]. Moreover, several types of malware were discovered that use DNS tunneling for establishing difficult to detect C&C channels for bot communications over the DNS protocol. The malware described in [29] and [30] uses DNS queries and responses to execute commands and control compromised machines over the DNS protocol. Therefore, securing them is essential.

2.3 Detection IDS/IPS systems

Intrusion Detection System and Intrusion Prevention System (IDS/IPS) appliances detect and/or mitigate Internet attacks including DNS tunneling attacks using signatures or by packet inspection. Farnham & Atlasis, 2013 described detection methods including attributes payload analysis and traffic analysis [9]. They also provided *Snort* custom rules based on entropy and statistical analysis. Like viruses which can be designed to avoid anti-viruses products by manipulating the virus behaviors and signatures, attackers also can deploy DNS tunneling attacks with previously unseen signatures to bypass IDS/IPS appliances. Packet inspection devices such as IPS's also have the ability to detect DNS tunneling attacks, but those products require high-speed computing machines to process the analysis which is expensive. Previous research proposed various approaches for detecting botnet and malware that uses the DNS protocol in their communications. The work in [31] conducted a classification of botnet detection techniques based on DNS traffic characteristics, including the honeynet-based and the IDS-based approaches. The first is usually deployed to collect information and analyzed it in order to understand the behavior and the attributes of botnets. The proposed system in [32] is an example of honeypot which is designed for tracking amplification DDoS attacks. They deployed 21 honeypots in different locations so that they can get more information about such an attack. Most of the attacks are shortlived, with most of the victims being attacked once. They also show that 96% of the attacks had the same source. In [33], Ramachandran et al. proposed a DNS-based blackhole list (DNSBL), which is an example of a botnet detection system based on a signature. In their paper, the authors developed techniques and heuristics that detects DNSBL reconnaissance activity whereby botmasters perform for looking up against the DNSBL to check whether their spamming bots have been blacklisted or not. Interestingly, as a result, they discovered bots were performing reconnaissance on IP addresses for bots in other botnets. In [34], Antonakakis et al. proposed the *Notos* system, which is a dynamic DNS reputation system. This system assumes that the activities of malicious DNS have different features and can be easily distinguished from benign DNS requests. The reputation score for a new domain is calculated based on building models of known legitimate domains and malicious domains, and they achieve this by collecting passive DNS query data and analyze the network and zone features of domains. The system has been tested and evaluated using a large ISP's network with DNS traffic from 1.4 million users. The Notos system has been highly accurate and achieves low false-positive rates. It has the ability to identify the new domain(s) before they even get released to the public blacklist. However, signature-based IDS systems have limitations. The proposed systems in [33, 34] and similar other systems are designed based on blacklists. Moreover, maintaining the blacklist update of known malicious addresses is not a simple task. Malware masters and hackers can easily avoid this kind of system by registering a new domain to perform their activities [31]. In [28], the authors introduced an issue of the data exfiltration over the DNS protocol. DNSxD is proposed to detect and mitigate against DNS attacks, including data exfiltration, tunneling, C&C communication, within the software-defined network (SDN) environment. Their approach to detecting the low-throughput data exfiltration consists of two techniques. Traffic analysis (TA) is based on volume and frequency of DNS requests within one-hour monitor windows. The deep packet inspection (DPI) technique relies on inspection of uncommon record types, query length, and query entropy based on threshold values. To evaluate DNSxD, the authors analyzed the popular DNS data exfiltration attacks and current exfiltration detection mechanisms within the Mininet network virtualization framework. The evaluation result demonstrated that the proposed application was able to detect exfiltration based on different attacking techniques. The big concern of this work is that DNSxD application was deployed within the SDN environment, which is vulnerable to various attacks such as DDoS attack that makes the network unavailable [22]. Next, the evaluation was performed using the Mininet network virtualization framework, which is a network emulator that creates a network of virtual hosts, switches, controllers, and links [35]. The authors in [36] demonstrated a method to deploy Splunk, which is a commercial software that is used to index a large dataset and provides keyword searching capabilities, dashboarding, reporting, and statistical analysis, within the network and walked through steps to install external modules for enabling the detection of DNS tunneling attacks.

2.4 Detection using Machine Learning

Machine learning is a method of data analysis that automates analytical model building. It is an important element of artificial intelligence that builds modules based on self-study from input data, and make decisions with minimal human intervention to solve a specific problem. There are three main categories of machine learning: supervised learning, unsupervised learning, and semi-supervised learning. Supervised learning uses labeled training data and algorithms for building a classifier to predict labels on new data. On the other hand, unsupervised machine learning uses unlabeled training data and constructs a classifier based on learning and analyzing the data attributes. Lastly, semi-supervised learning is a classifier that uses both labled and unlabled data to train the algorithm [14]. In [14, 37], detection approaches based on DNS traffic are introduced using supervised machine learning. Hoang and Nguyen, 2018 evaluate the effectiveness of supervised learning techniques in botnet detection based on DNS traffic, including k-nearest neighbor (KNN), decision tree, random forest, naive Bayes. The evaluation result shows that the machine learning algorithms produced a high detection accuracy of 90%, and they can be used effectively in implementing a botnet detection system. Machine learning is being used for detecting various security threats including DNS tunneling, data exfiltration attacks and others. In [6], the authors proposed an anomaly-based intrusion detection system using behavior analysis approach to protect against cyber attacks. Using supervised machine learning, the proposed method can distinguish abnormal behavior and detects the known and unknown DNS attacks with a low false-positive rate. In their experimental results, however, the authors did not provide enough information about the attack's data set. Liu et al., 2017 proposed a prototype system that detects a DNS tunneling attack using binary-classification based on four behavior features including time-interval, request packet size, record type, and subdomain entropy features [38]. Their results show that their system achieved a high rate of detection accuracy of 99.9%. However, the system was trained to detect only well-known tools which are based on expected behavior, and an attacker can bypass the system by manipulating these features. The authors in [39] developed a machine learning method of detecting DNS tunneling attacks including information exfiltration from compromised machines and DNS tunneling that was used to establish C&C servers. Their detection method relied on detecting DNS tunneling attacks that use *TXT* records only. However, adversaries can bypass the detection of this method by performing the attack using different techniques and DNS records [27]. In [9], the authors discussed DNS tunneling attacks and its tools that use common and uncommon types of DNS records such as *A*, *AAAA*, *CNAME*, *NS*, *TXT*, *MX*, *SRV* and *NULL* [27]. Nadler et al., 2019 proposed a novel approach for detecting and denying both DNS tunneling attacks and low throughput data exfiltration over the DNS using machine learning techniques [2]. There are three main phases for performing their approach. The initial phase starts by collecting and grouping the DNS data log per domain name for an extended period of time. The next phase is extracting features based on querying behavior of each domain name. Finally, a classification phase, which is conducted using the isolation forest algorithm, is used to determine whether the domain name was used for data exfiltration. Based on the classification model results, DNS queries and responses from or to these malicious domains will be blocked. They performed the approach on large-scale DNS traffic, DNS tunneling tools traffic, well-known malware traffic, and sensitive data that was injected into the dataset to evaluate their work. As a result, the system detects the attacks with a false positive rate of less than 0.002% [2]. However, the proposed method does not work in real-time, requiring time for collecting data to detect the low throughput malware traffic. During the collection phases, an attacker can use the DNS exfiltration technique and transfer sensitive data over the DNS.

2.5 Detection using Visualization

(Choi, Lee, & Kim, 2009) developed a comprehensive and useful tool for visualizing and detecting network attacks using the parallel coordinates technique [13]. Nine signatures were defined, one for each attack [13]. The types of attack included portscan, hostscan, worm, source-spoofed denial-of-service (DoS), backscatter, and network and signal DoS attacks. In addition, they developed a detection mechanism based on a hash algorithm. However, the authors did not include DNS tunneling as one of the Internet attacks. In [10], the authors focused on visualizing network attacks such as botnets that cause various types of attacks such as DDoS, spam, and others. The visualization system was developed to provide visual information which helps in detecting botnet attacks using DNS traffic. The parallel coordinates technique, a common way of visualizing high-dimensional geometry and analyzing big and multi-dimensional data, was used to represent three different DNS parameters in real-time. Four graphical patterns were defined in that visualization system [10, 11]. In [40], a fast and scalable visualization system for detecting and identifying DNS tunneling attacks is proposed based on the parallel coordinates technique. The visualization system was developed to provide visual information which helps in detecting botnet attacks using DNS traffic. One week of data was analyzed to study the normal behavior of DNS, and it was compared to the behavior during a tunneling attack. Six DNS parameters for the parallel coordinates technique were identified and used to visualize the DNS traffic using the parallel coordinates technique in order to distinguish between regular DNS traffic and malicious traffic containing DNS tunneling attacks. Four different DNS tunneling patterns were identified, each representing different types of DNS tunneling attacks using the parallel coordinates technique [40]. In [41], Born and Gustafson (2010) introduced the NgViz tool that combines a visualization technique with N-gram analysis to examine and show DNS traffic from any suspicious activities. N-gram is used in text mining and natural language for processing tasks such as predicting the next item in a sequence. Their method compared a given input file of domain names with fingerprints of legitimate traffic. It is important to note that in this research, the authors supplied the comparison input of domain names and fingerprint of legitimate files manually.

2.6 DNSSec

To overcome some of the security issues related to the DNS protocol, DNS has had many improvements over the years, and there are significant changes in the protocol were made, such as the randomization of the source port and transaction ID to prevent the DNS cache poisoning attack, which redirects network clients to malicious servers. Moreover, Eastlake and Kaufman (1997) implemented DNSSec by the Internet Engineering Task Force [42]. DNSSec is an additional security layer added to the existing DNS protocol. The primary concept of designing it was based on the *chain of trust* that applies the concept of public key cryptography for achieving origin authentication and data integrity on the replied

DNS requests and verifying the *non-existence* of a domain, which makes DNSSEC effective against types of cache-poisoning attacks [43]. However, some of its limitations have been discussed in [44]. This solution is not sufficient for every network because DNSSec increases the amount of network traffic. It also comes with other issues such as broken validations which may cause a denial of service.

Even though some research has been conducted regarding approaches for detecting and identifying the DNS tunneling attack, some issues still deserve further attention. IDS and IPS detection techniques are based on pre-defined signatures by observing events and identifying patterns which can be applied on the signatures of known tools and attacks. One of the presumptions of the machine learning methods used for detecting malicious activities in traffic is to compare the characteristics of traffic with DNS tunneling with the normal DNS traffic even though some of the well-known tools are generating tunneled traffic similar as possible to the regular DNS queries [38].

The next chapter describes the methodology and evaluation of the work. It includes a description of the visualization module and how graphical representations of different DNS tunneling attacks are generated, a description of the detection module, and a description of how the system will be evaluated and metrics to measure the performance of the system. Finally, it provides initial results of the visualization module.

3 Visualization of DNS Tunneling Attacks

The Domain Name System (DNS) is considered one of the most critical protocols on the Internet. The DNS translates readable domain names into Internet Protocol (IP) addresses and vice-versa. The DNS tunneling attack uses DNS to create a covert channel for bypassing the firewall and performing command and control functions from within a compromised network or to transfer data to and from the network. There is work for detecting attacks that use DNS but little work focusing on the DNS tunneling attack. In this chapter, we introduce a fast and scalable approach, using the parallel coordinates technique, visualizing a malicious DNS tunneling attack within the large amount of network traffic. The DNS tunneling attack was performed in order to study the differences between the normal and the malicious traffic. Based on different scenarios, four different DNS tunneling graphical patterns were defined for distinguishing between normal DNS traffic and malicious traffic containing DNS tunneling attacks. Finally, the visualization system was able to visualize the DNS tunneling attack efficiently for the future work of creating an efficient detection system.

3.1 Introduction

Visualization is not a new concept for monitoring network traffic. There are several papers that used the parallel coordinates technique to monitor and visualize network traffic. There are various benefits of using a parallel coordinates technique. First, it is scalable. It helps to represent a considerable amount of data in an efficient manner which makes analyzing big data more simple and powerful. Second, there is no limit on the number of values for representing an attack using the parallel coordinates technique that converts multidimensional data into two dimensions. Therefore, using the parallel coordinates technique gives the ability to represent more than three different values within a two-dimensional space. In this way, the relationship between the results can be reviewed quickly showing prominent trends, correlations, and divergence from the raw data. Finally, the visualized data using the parallel coordinates technique does not include any bias for any column within the space [12, 13]. In other words, every visualized feature has the same weight.

The contributions of this chapter are: (1) A fast and scalable visualization system is proposed for recognizing DNS tunneling attacks using the parallel coordinates technique. The proposed system is able to distinguish between normal and malicious DNS traffic. (2) A real-world DNS tunneling attack is conducted within our network to study the behavior of the attack traffic. Four different graphical patterns are defined in this work by analyzing the DNS tunneling traffic based on different scenarios.

3.2 Visualization System Design

In this section, we describe the system architecture design of a visualization system for visualizing and identifying the Domain Name System (DNS) tunneling attacks using attack patterns as shown in Figure 3.1. The visualization system consists of three main components: Parser, Analyzer, and Visualizer. DNS data can be passively captured at the network edge or it could be read from existing PCAP files. The Parsing was coded in the Go programming language that interfaces with the TCPDUMP library for collecting the data quickly and efficiently. The captured packets are filtered to provide all User Datagram Protocol (UDP) packets on port 53. Note that DNS can also use TCP port 53 for things such as zone transfers but it was decided to only capture the more common DNS traffic that uses UDP. Performance is one of the essential factors necessary for this system. Therefore, the focus is on the DNS response data sent by the DNS server that confirms the communications between the local computer and the local or global DNS servers. Another reason why DNS response packets were considered, is that it decreases the number of captured packets, resulting in faster computations. The DNS response packets have a significant amount of information. Therefore, it was found that a subset of information could be used to visualize and identify DNS tunneling attacks and the parameters include the ID, domain and subdomain names, TTL (Time To Live), source and destination IP, date and time. This DNS data is stored in a database or file as input into the Analyzer.

Next, the Analyzer retrieves the captured DNS data from the database. First, it checks that the captured data is a proper DNS packet to avoid further issues. Then, the Analyzer filters the data to eliminate duplicate records based on the ID number and it applies analytical and mathematical functions on the captured data. For instance, it gets the number of requests for each domain name based on date and hours and sorts them by decreasing values. It selects the top five domain names that have the most frequent requests, and it compares them to the value of the threshold. If any of these values exceed the threshold,

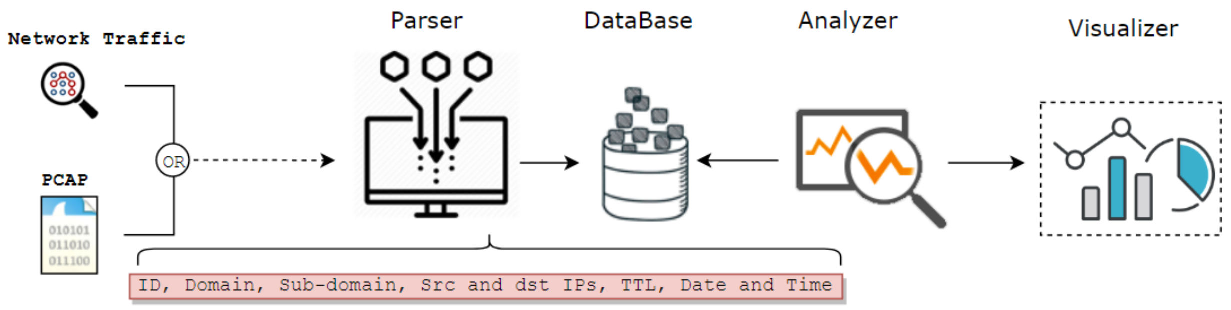


Figure 3.1: System Architecture Design

then it considers it as a thread. The Analyzer module sends the suspicious domain names to the Visualizer.

Finally, the Visualizer displays the analyzed data using the parallel coordinates technique. This technique uses each coordinate to represent the six different parameters: source and destination IP address, domain and subdomain names, TTL, and the time. For example, assume that a workstation with *192.168.0.1* IP address sends a DNS request to the global google DNS server that has *8.8.8.8* IP address to resolve the following domain name *www.attacker.com.* Then, the Visualizer module represents the source coordinate with the local workstation IP address *192.168.0.1*, the destination coordinate as *8.8.8.8*, *attacker* as the domain name value, and *www* as a subdomain name value. If we have thousands of requests, the visualization system displays a funnel-like pattern.

3.3 Graphical patterns for DNS tunneling attacks

In this section, the parallel coordinates technique is introduced that uses the captured data to represent the DNS tunneling as graphical patterns. Four DNS tunneling attack patterns are identified using the parallel coordinates technique, which can be used to both identify and understand DNS tunneling attacks. Several tunneling tools were found that implement the DNS tunneling attack, such as *iodine*, *dns2tcp*, *DeNiSe*, *dnscapy*, *heyoka*, *fraud-bridge*, *dnscat*, and *dnscat2*. Each DNS tunneling tool uses a slightly different implementation of the attack. For example, *dnscat2* uses a *NULL* for a particular request type, while *heyoka* uses different encoding techniques to pass the data over the DNS communication channel [45]. However, all of these tools are using the same concept. Note that the DNS protocol is not able to transfer large amounts data, only small to average amounts.

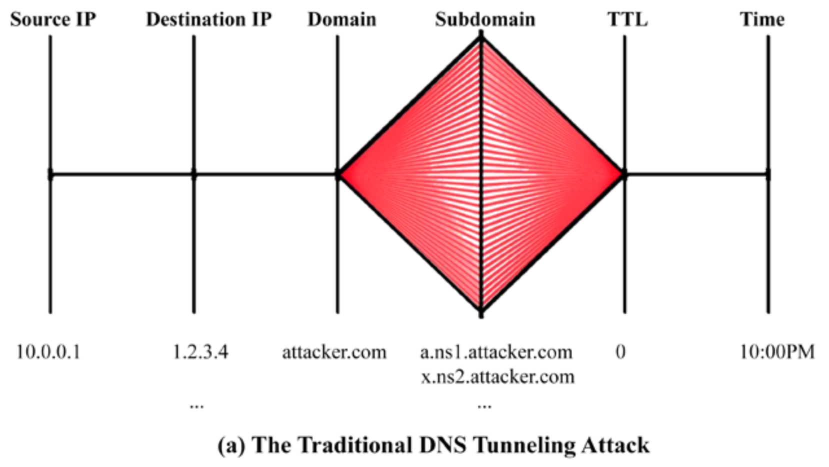
Therefore, the DNS tunneling attack requires sending a considerable number of queries in order to transfer the data and this leads to the ability to identify these types of attacks.

In the parallel coordinates technique, each coordinate represents a particular value. The first coordinate is the source IP address which has the value of the computer that receives the DNS query while the second coordinate is the destination IP address that forwards the DNS request. The third coordinate represents the domain names, and the fourth coordinate contains subdomain names. Then, the fifth coordinate is the TTL value that is set by the Authority server which allows a DNS client to cache the answer for a particular time, while the time is the last coordinate. With these values, our visualization system shows a graphical pattern that can be matched to a template of the predefined graphical patterns of unusual DNS traffic.

Suppose that an attacker compromised a computer within a network and he needs to send stolen data out secretly without being detected by the firewall. The attacker can use a DNS tunneling attack to hide his movements or to transfer data outside the network. In this case, the attacker requires a pre-configured fake DNS server outside the network. From inside the compromised network, he can connect to his fake server via the DNS protocol that is allowed by the firewall in most networks. During the DNS tunneling attack, it sends a tremendous number of requests to the DNS server within a short amount of time actually transferring data instead of just sending DNS requests. In this work, we ran the DNS tunneling attack inside our network and we transferred a 12 MB text file. During one DNS tunneling attack, it generated a considerable number of DNS requests (52 thousand queries), which is significantly more than normal. This kind of attack splits the file into small pieces and encodes each piece with a specific encoding technique [9]. Then, it encapsulates the pieces into the DNS packets to transfer it to the preconfigured fake DNS server that has multiple preconfigured subdomain names.

The parallel coordinates visualization technique captures this unusual activity by generating attack signatures using the number of requests for each parameter. For example, Figure 3.2.A exhibits the previous scenario of the traditional DNS tunneling attack during one hour where the source coordinate represents the value of the attacker's computer which is *192.168.0.1*. On the destination coordinate, we can see that it has the value of global DNS server and one domain name in the domain coordinate. However, we observe more than 52 thousand subdomain names, which shows an unusual number of DNS requests within a short time. One of DNS tunneling features is that the fake DNS server sends thousands of

DNS requests within a short time where it has a constant TTL value. The previous example shows all 52 thousand requests have a TTL value of zero. Figure 3.2.B or malware scenarios using the traditional DNS attack tunneling technique to hide their C&C communications inside the network. It shows that we have many internal computers sending and receiving many DNS requests within a short time. In [45], they presented the Heyoka tool which is one



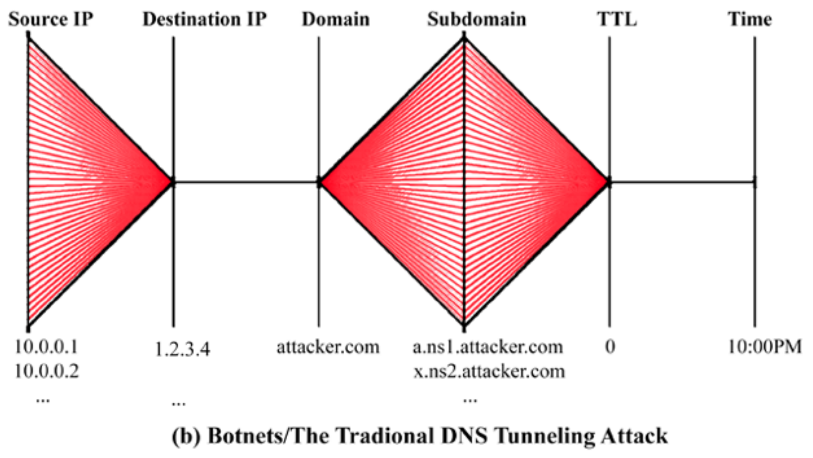
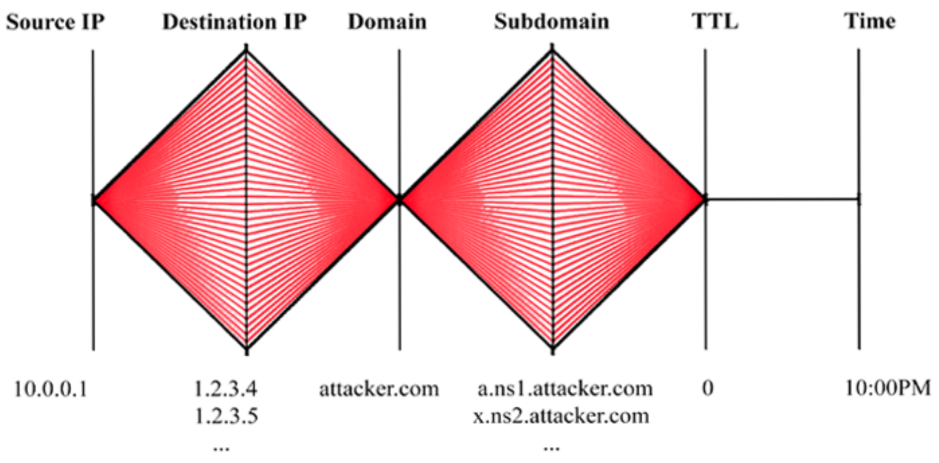
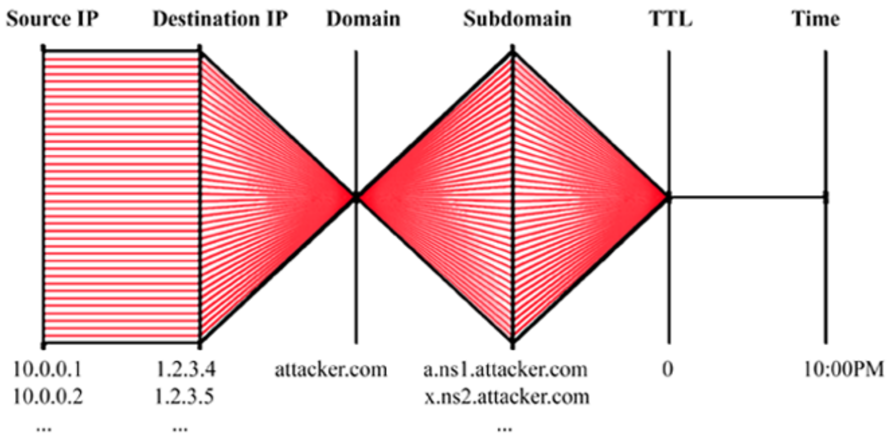


Figure 3.2: Graphical Patterns for Traditional DNS Tunneling Attack.

of the DNS tunneling tools that not only performs the attack but it also has some unique features that are not implemented in other DNS tunneling tools. Using Heyoka, an attacker can deploy a master DNS server with multiple slave DNS servers. One could use a different subdomain for each slave, crafting requests like the following: encoded-data.n.attacker.com, where n is the slave identifier number. Therefore, Figure 3.3.C shows this case when the attacker uses a multiple slave DNS server while Figure 3.3.D shows the case of using the



(c) Heyoka DNS Tunneling Attack



(d) Botnet/Heyoka DNS Tunneling Attack

Figure 3.3: Graphical Patterns for Heyoka DNS Tunneling Attack.

Heyoka tool when it has been used by botnet clients or malware situations from multiple computers.

3.4 Evaluation

3.4.1 Experimental setup

In this section, we show our experimental results of the experiments we performed to evaluate the visualization system. The evaluation method consists of two main parts. The first part is performing the DNS tunneling attack. The second part is importing the captured DNS traffic data to the visualization system.

In order to perform DNS tunneling attack, we register a domain name and set up an

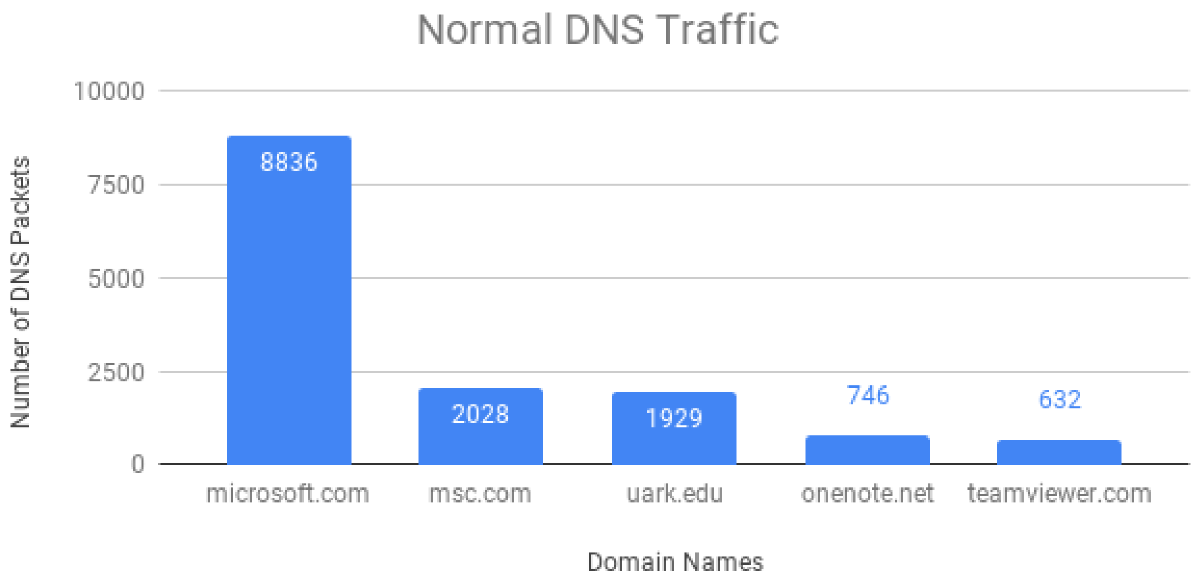


Figure 3.4: Top Five Domain Name for an Entire Week without the Attack.

external server with a static IP address. Debian Linux server was installed and set up in the cloud for this purpose. The DNS tunneling attack was performed using Kali Linux in our internal network to encapsulate and upload a 12 MB text file to an external FTP server during a weekday. Moreover, we also sniffed and monitored normal DNS traffic during regular daily use for an entire week. There are two significant reasons for sniffing and capturing a regular DNS traffic in our experiment. The first reason is to compare and study the normal behavior of DNS traffic against the DNS traffic that has the DNS tunneling attack while the second reason is to determine and set the value of threshold based on the normal DNS traffic level.

3.4.2 Results

After performing the attack, the PCAP file was stored for the later analysis. Next, using the Analyzer module, the captured data was analyzed excluding the DNS tunneling attack as shown in Figure 3.4. We found that *Microsoft.com* is the value that occurs most frequently in the data set, having 8836 DNS requests for an entire week and 170 requests per an hour as a maximum value.

As shown in Figure 3.5, when implementing the DNS tunneling attack, the malicious domain name has the most frequently occurring value with more than 50000 DNS requests and 40993 requests per an hour as a maximum value. It is imperative to note that the attack has run for only two hours. This experiment was used to empirically determine the threshold. In this environment, the sum of the number of requests of all domain names was 489 per hour. Therefore, the value of the threshold has been set to 500 DNS requests in our experiment. It is important to note that the threshold has to be set appropriately for each network depending upon the typical traffic levels.

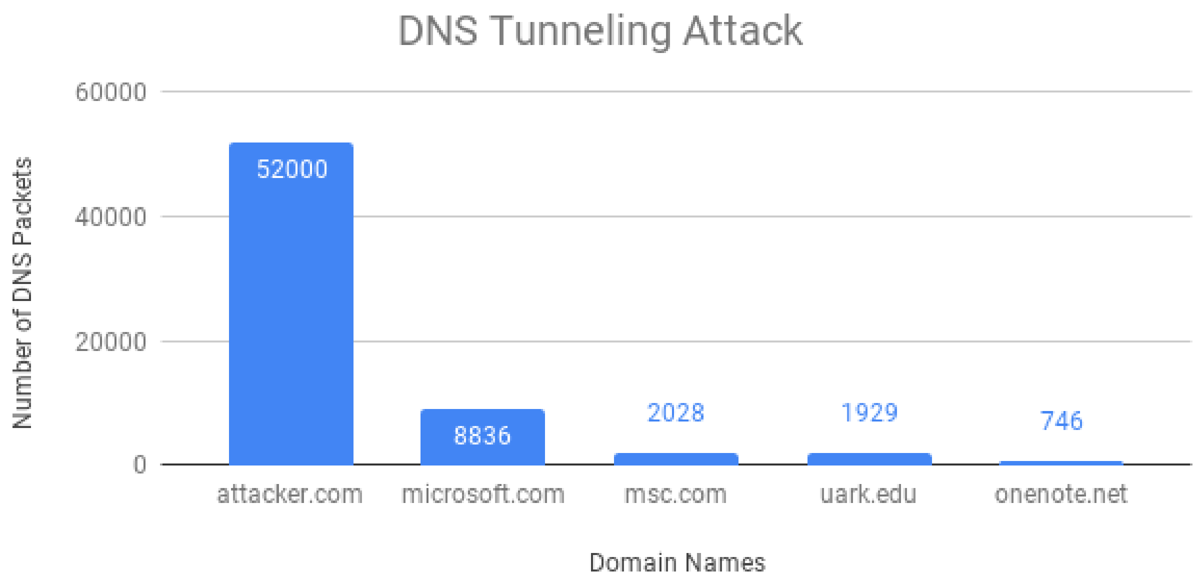


Figure 3.5: Top Five Domain Names during the Attack over a Two-hour Period.

Next, the malicious domain name is sent to the Visualizer module to represent a possible attack using the parallel coordinates technique. Figure 3.7 shows the visualization of the entire captured network traffic that contains a DNS tunneling attack without the thresholding technique turned on while Figure 3.6 shows the visualization of the network traffic with the DNS tunneling attack with the thresholding technique enabled. As seen in the figures, there is an obvious funnel-like pattern that appears during the attack with the proposed technique. As shown in Figure 3.7, there are four values on the source IP address coordinate which represent the internal computers where they requested too many domains and subdomain names during the day. On the other hand, Figure 3.6 shows that only one computer in the internal network sent too many DNS requests to resolve a considerable number of subdomain names of an attacker domain name within two hours.

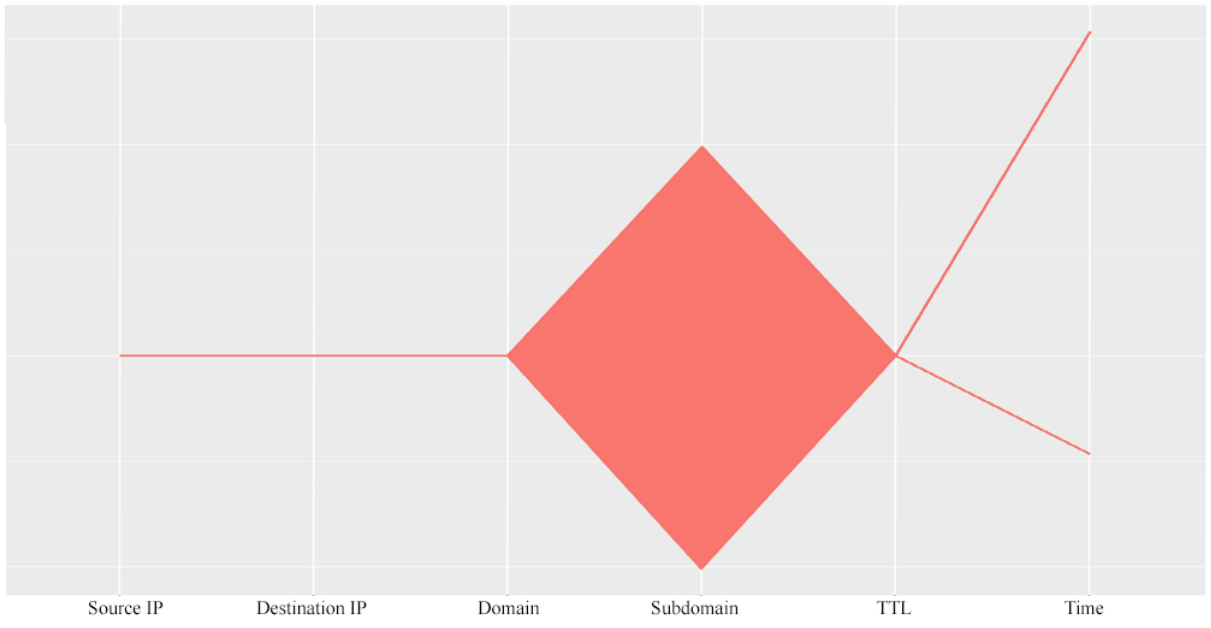


Figure 3.6: Visualizer Module Shows only the Attack.

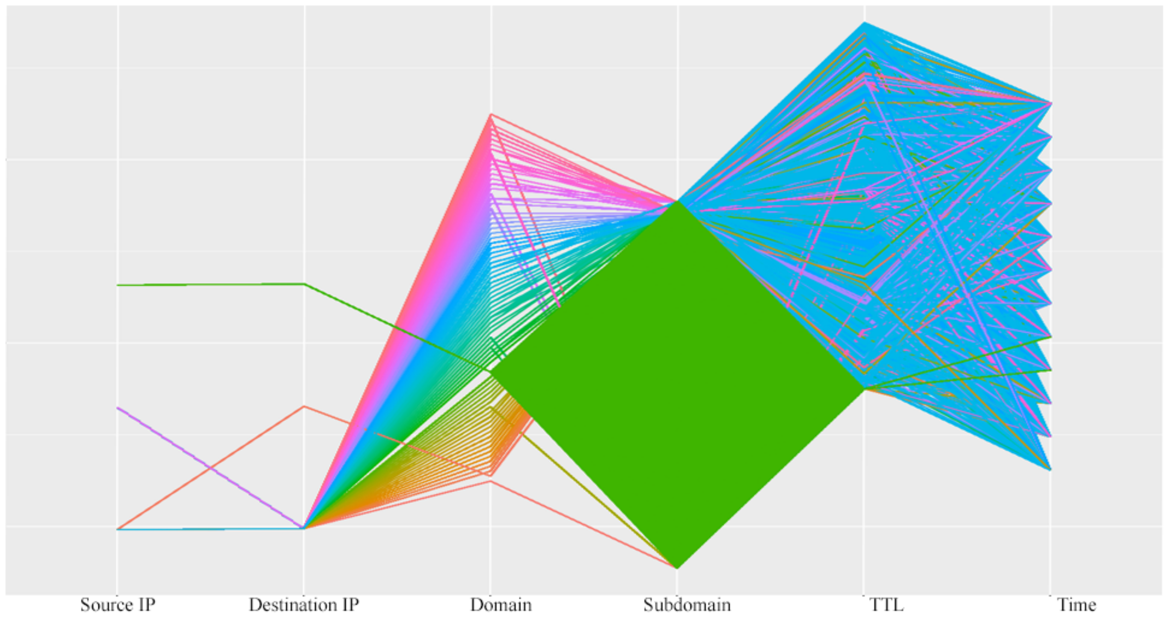


Figure 3.7: Visualize the Entire DNS Captured Data.

3.4.3 Issues

Several issues occurred during the initial design of the visualization system. First, the entire visualization system had been coded using Python and Pandas which is the Python Data Analysis Library [46]. Results were satisfying with a small number of data records but not a large number. As soon as our data set became larger, the system was not able to process the data as efficiently resulting in slower turnaround times. The Parser module was not able to process big PCAP files and a large number of DNS packets, and the Visualizer module was taking more than 16 hours of processing time to analyze and display the data on the parallel coordinates. Therefore, the system was rewritten in the Go Programming Language [47] and the R Language [48]. After rewriting the code, 5GB PCAP files with more than 52000 DNS packets are parsed in less than 12 seconds, while R processes and presents the parsed data in approximately 5 seconds.

3.5 Conclusion

In this chapter, a fast and scalable visualization system for detecting and identifying DNS tunneling attacks is introduced based on the parallel coordinates technique. To the best of our knowledge, this is the first time that the DNS tunneling attack has been visualized. Data was collected and visualized from an internal network during a DNS tunneling attack. One week of data was analyzed to study the normal behavior of DNS and it was compared to the behavior during a tunneling attack. In addition, based on our analysis, a method is provided to set the threshold value. Six DNS parameters for the parallel coordinates technique were identified and used to visualize the DNS traffic using the parallel coordinates technique in order to distinguish between normal DNS traffic and malicious traffic containing DNS tunneling attacks. Four different DNS tunneling patterns were identified each representing different types of DNS tunneling attacks using the parallel coordinates technique. Visualizations were created that are obviously very different so that they can be used in the future for detecting DNS tunneling attacks.

4 DNS Tunneling Attack Detection using Machine Learning

Individuals and organizations rely on the Internet as an essential environment for personal or business transactions. However, individuals and organizations have been primary targets for cyber-security attacks that steal sensitive data. Adversaries can use different approaches to hide their activities inside the compromised network and communicate covertly between the malicious servers and the victims. The domain name system (DNS) protocol is one of these approaches that adversaries use to transfer stolen data outside the organization's network using various forms of DNS tunneling attacks. DNS protocol is considered one of the most important internet protocol because it is available in almost every network, ignored, and rarely monitored. In this work, our research focuses on implementing a reliable and robust detection system against DNS tunneling attacks using visualization and machine learning techniques. The detection system is able to detect DNS tunneling attacks, which are either embedded in malware or be used as stand-alone attacking tools. The system consists of three main modules: the Parser, the Analyzer, and the Detector. The visualization module's output will be input for the developed detection module that will use machine learning algorithms to decide whether a DNS tunneling attack occurs. The system is trained, tested, and evaluated using metrics derived from a confusion matrix that is commonly used to measure intrusion detection systems' performance. It is shown that the system is capable of detecting the DNS tunneling attack with a low false rate and high accuracy. The results show that all of the machine learning algorithms tested, including Decision Tree, Logistic Regression, Naïve Bayes, K-Nearest Neighbors, Random Forest, and Support Vector Machine classifiers, used in the experiment achieved 99% accuracy to identify the attack.

4.1 Methodology

4.1.1 System Design

The proposed detection system extends the work done in [40]. This work uses the parallel coordinates technique to confirm the hypothesis that the visualization of DNS network traffic using the parallel coordinates technique with an active DNS tunneling attack is visually different from DNS network traffic that does not contain the attack. The visualization system was developed to provide visual information for detecting botnet attacks using DNS traffic for communication. Six DNS features are used as parameters for the parallel coordinates technique to identify and visualize the DNS traffic to distinguish between regular DNS traffic and malicious traffic containing DNS tunneling attacks. Four different DNS tunneling patterns were identified, each representing various types of DNS tunneling attacks [40]. The DNS tunneling attack signatures were derived from the work in [40] to implement the detection system to label and prepare the machine language algorithms' dataset. Also, we evaluate the detection system on a dataset that was generated within the lab. The evaluation method consists of two main parts. The first part performs the DNS tunneling attack to create a dataset, and the second part imports the captured DNS traffic dataset to the detection system. We employ the four types of attacks in our detection system and generate relationship signatures for each one to detect and flag the attacks. Figure 1.4 shows the original visualizations and Table 5.1 shows the attack relationship signatures. During the analysis process, we used these signatures to label the data for training the machine learning algorithms to detect the DNS tunneling attack.

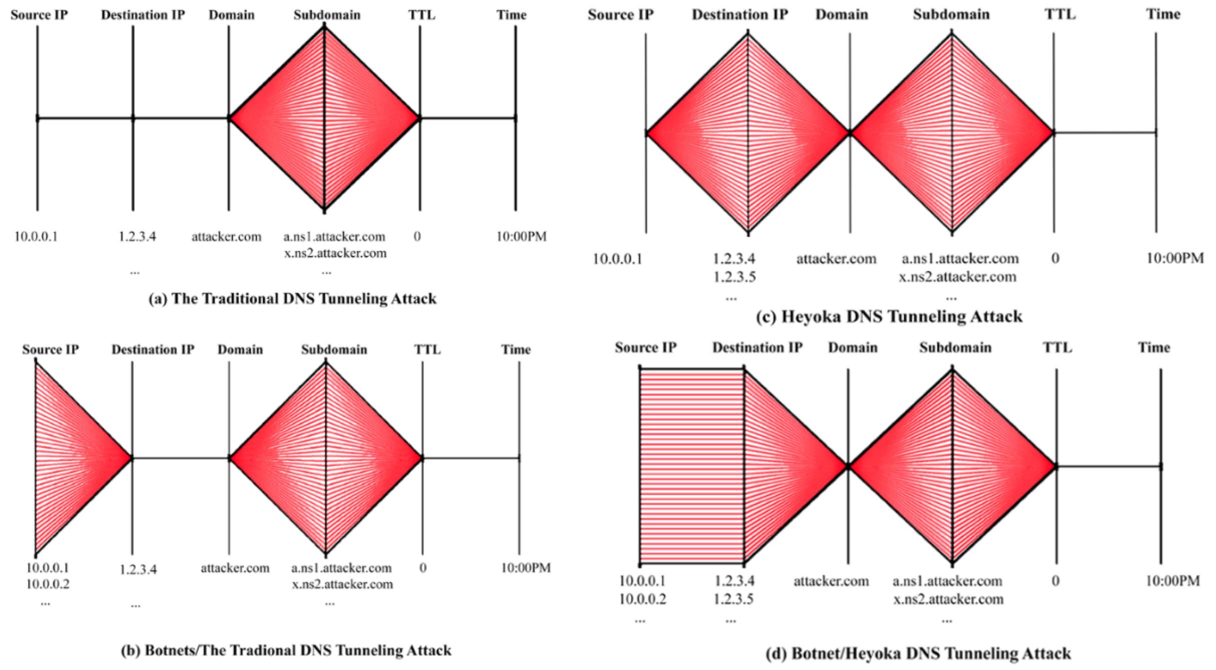

Source IP
Destination IP
Domain
Subdomain
TTL
Time
10.0.0.1
1.2.3.4
attacker.com
a.ns1.attacker.com
x.ns2.attacker.com
0
10:00PM
(a) The Traditional DNS Tunneling Attack
1.2.3.5
(c) Heyoka DNS Tunneling Attack
10.0.0.2
(b) Botnets/The Tradional DNS Tunneling Attack
(d) Botnet/Heyoka DNS Tunneling Attack

Figure 4.1: The DNS Tunneling Attack Patterns

The visualization of the traditional DNS tunneling attack shown in Figure 1.4, plots the quantity of source IP addresses, destination IP addresses, domain names, subdomain names, time-to-live (TTL) values, and time values. There is an easily identifiable pattern that is converted to a signature represented as the tuple *(1, 1, 1, m, 1, 1)*, which means there is 1 source IP address, 1 destination IP address, 1 domain name, multiple (m) subdomains, a single TTL, and a single or small window of time. The traditional DNS tunneling attack, Botnet DNS tunneling attack, Heyoka DNS tunneling attack, and Heyoka/Botnet DNS tunneling attack all have unique visualizations that were converted to attack relationship signatures.

Table 4.1: DNS Tunneling Visualization Signatures

Attack	Signature	Representation	Normal\Malicious
Traditional D. T.	1,1,1,m,1,1	Attack_Type1	True
Botnet D. T.	m,1,1,m,1,1	Attack_Type2	True
Heyoka D. T.	1,m,1,m,1,1	Attack_Type3	True
Heyoka /Botnet D. T.	m,m,1,m,1,1	Attack_Type4	True
No Attack	Not Above	0	False

4.1.2 The Detection System

In this section, we discuss the design of the detection system. The detection system consists of three main modules: The Parser, the Analyzer, and the Detector. DNS data can be passively captured at the network edge or it could be read from existing PCAP files via the Parser. The main goal of the Parser is to capture and collect a subset of DNS information on the network edge and store it in the database. The Parsing module is coded in the Go programming language that interfaces with the TCPDUMP library for collecting the data quickly and efficiently. The raw captured packets are filtered to provide all User Datagram Protocol (UDP) packets on port 53. Note that DNS can also use TCP port 53 for things such as zone transfers but it was decided to only capture the more common DNS traffic that uses UDP. Performance is one of the essential factors necessary for this system. Therefore, the focus is on the DNS response data sent by the DNS server to the local computer. Another reason why DNS response packets were considered, is that it decreases the number of captured packets, resulting in faster computations. The DNS response packets have a significant amount of information. It was found that a subset of information could be used in identifying DNS tunneling attacks and the DNS features include the ID, domain and subdomain names, DNS record type, TTL (Time-to-Live), source and destination IP, SRV's IP (Service IP), data length, and finally the date and time. These DNS features are stored in a database or file as input for the next module which is the Analyzer module.

The next module is the Analyzer module, which cleans and prepares the DNS captured data and then passes it to the next module. The Analyzer retrieves the captured DNS data from the database, and it validates the data by checking the captured data to verify that it is a proper DNS packet to avoid further issues. Then, the Analyzer module filters the data to eliminate duplicate records based on the ID number and it applies analytical and mathematical functions on the captured data. Examples of analytical and mathematical function are calculating the number of requests for each domain and its unique subdomain names, the number of source IPs per domain, the number of destinations of IP per domain, the number of SRV of IPs per domain, the average number of subdomain lengths per domain, the average number of data lengths of each DNS request per domain, and the average number of TTLs per domain. The Analyzer module also generates the attack signature for each domain and compares it with the malicious attack signatures to label the domain name as malicious or not. Finally, the Analyzer module prepares the analyzed data and stores this information into a file to pass to the machine learning algorithms to determine if there any malicious domain names. Also, the Alexa and umbrella top 1000 domains [49] are used to identify and label the captured domain name as legitimate.

Finally, the Detector module retrieves the prepared data and applies the machine learning algorithms on the data to determine whether the DNS tunneling attack is occurring or not. After retrieving the data, the process of this module is to prepare and split the DNS data into two datasets: the first dataset is for training while the other one is for testing the machine learning algorithms. Then, we train the machine learning algorithms on the dataset, which has the labels, and finally, we examine the machine learning algorithms on the dataset. The machine learning algorithms used in this work are the decision tree classifier, naïve Bayes classifier, K-nearest-Neighbors classifier, Logistic Regression classifiers, and Random Forest classifier.

4.2 Results and Discussion

4.2.1 Evaluation

In this section, we discuss the evaluation method for the DNS tunneling detection system. To evaluate the performance of the methodology, we created the normal DNS dataset. Three months of DNS traffic were captured within the network. The network environment is a home-based lab, consisting of more than 20 devices, including IoTs, servers, mobiles, PCs, and laptops. There are two significant reasons for capturing regular DNS traffic in the experiment. The first reason is to compare and study the normal behavior of DNS traffic against the DNS traffic that has the DNS tunneling attack while the second reason is to determine and set the value of the threshold based on the normal DNS traffic level.

Due to the importance of the DNS protocol, DNS tunneling tools were used to simulate the tunneling attacks using various methods. For example, the *dns2tcp* DNS tunneling tool uses the *TXT* record types to perform the tunneling, whereas *Iodine* DNS tunneling tool uses *NULL* records [50]. However, due to the lack of available DNS tunneling datasets, we implemented and created a DNS dataset that has malicious DNS traffic. The DNS tunneling dataset includes various DNS traffic generated from DNS tunneling tools, including *Iodine*, *DNScat2*, *dns2tcp*, and *fraud-bridge*. Finally, normal and malicious DNS datasets are combined and then imported to the detection system and evaluated.

4.2.2 Experimental Setup

This work is a network-based solution which has the ability to distinguish and identify between normal and malicious DNS traffic. In order to detect the malicious DNS traffic, we require that the system intercepts the network DNS traffic. To guarantee that the system captures all transmission DNS traffic, the plan is to place the system on the network edge or to run it as a DNS proxy, which looks up DNS queries from a local cache. A DNS proxy makes the process of resolving the domain name more efficient by performing the lookup on behalf of the client and caching common queries. In addition, the detection system will have the ability to read DNS traffic from an existing PCAP file, which helps to identify and detect the attack offline from previously captured network traffic.

The DNS tunneling attacks signatures rely on the work that is done in [40], which uses the parallel coordinates visualization technique to detect and visualize the DNS tunneling attacks. The visualization system was developed to provide visual information that helps in detecting botnet attacks that are using DNS traffic for communication. The evaluation process relies on two parts including: create the dataset that has the DNS tunneling attacks and import the generated dataset into the detection system.

To perform a DNS tunneling attack, we register a domain name and set up an external server with a static IP address. A Debian Linux server is deployed in the cloud for this purpose. The DNS tunneling attack is performed using the Kali Linux distribution that has many DNS tunneling attack tools built-in on the internal network to encapsulate and upload a 12 MB text file to an external FTP server. Also, we configured a couple of local domain nameservers to perform other DNS tunneling attacks to generate more data for our evaluation. During the attacks, we captured DNS traffic as PCAP files. As previously mentioned, we captured the normal DNS traffic on a three-month window to generate our dataset for the evaluation process.

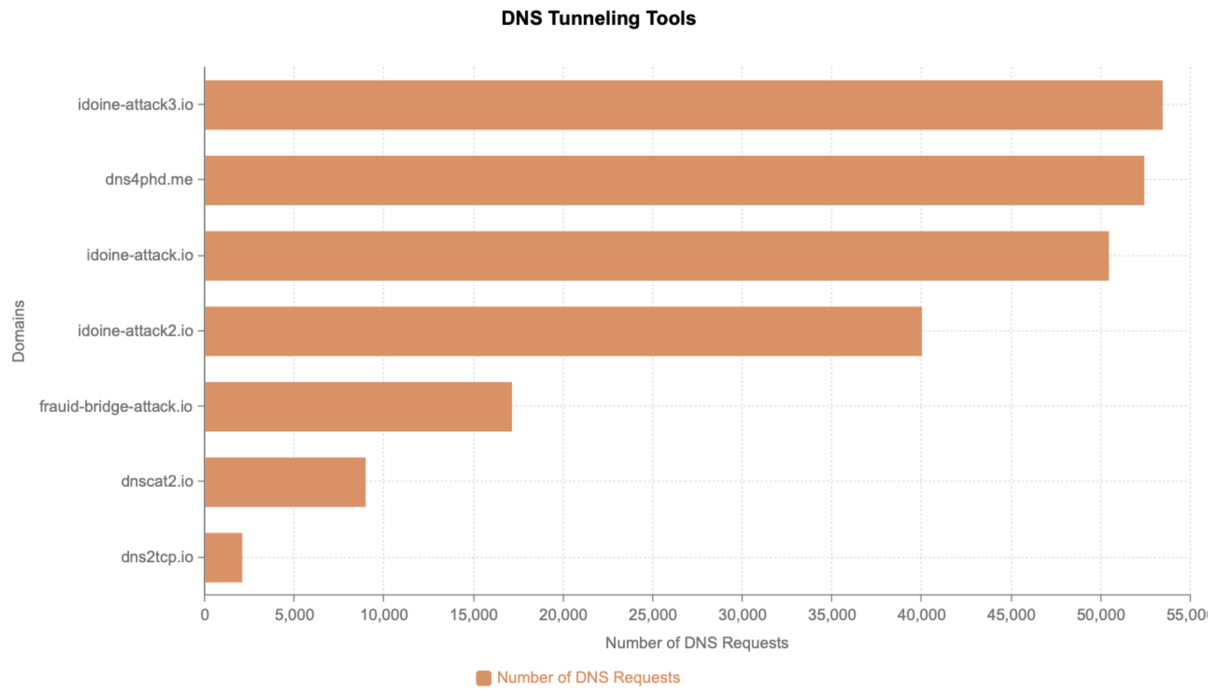


Figure 4.2: DNS Tunneling Tools

As shown in Fig 4.2, the five DNS tunneling attack tools that implement seven different types of tunneling attacks that were used in our research to generate and simulate the malicious DNS traffic. Each DNS tunneling attack tool is performed for a specific time. Figure 4.2 shows that the *iodine-attack3.io* domain name has the most frequency of DNS requests among all the attacks, and that is because it was executed twice during the three-month window period. On the other hand, the number of requests to the *dns4phd.me* domain name is performed a couple of hours during the attack to upload a text file via an FTP server over the Internet while the other tools performed within a short time so that they generated fewer DNS requests within the home-lab environment.

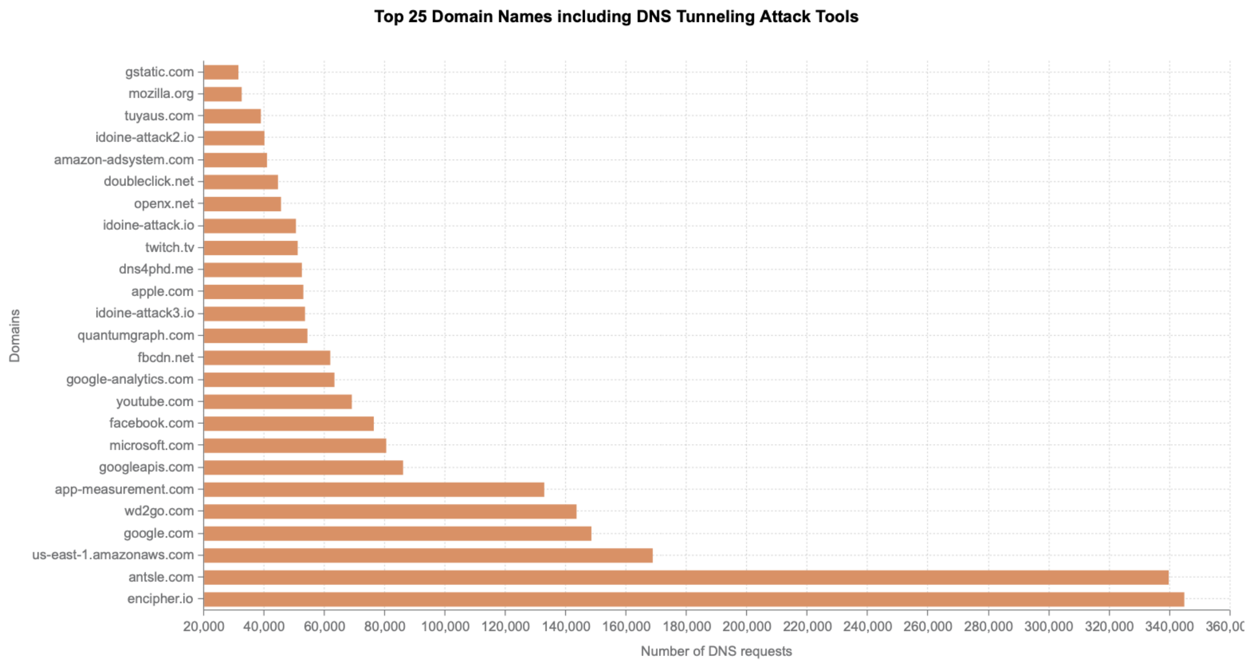


Figure 4.3: Top 25 Domain Names including DNS Tunneling Attack Tools

Figure 4.3 shows the top 25 domain names for the three-month window period, including the DNS tunneling attacks. Note that the malicious domain names, which are *iodine-attack3.io* ,*dns4phd.me*, *iodine-attack.io*, and *iodine-attack2.io*, are including the list of the top 25 domain names of the most frequency DNS request for the three-month window period of capturing the DNS traffic. There are two methods that the detection system feeds the machine learning algorithms with data include, capturing live network traffic and importing PCAP files. During the evaluation, we used the PCAP file, which was generated before, to import the data to the detection system. As previously mentioned, the PCAP file has been captured data for a three-month window period. Note that we imported the complete PCAP file at once to the detection system. However, our detection system is intended to capture and import DNS requests for a shorter time in the future, making detecting the DNS tunneling attack easier. For example, the detection system captures data every five minutes and analyzes it to identify any malicious DNS traffic, which makes it a more efficient method.

4.2.3 The Dataset

After performing the attacks, the PCAP files were stored for the later analysis. Next, using the Analyzer module, the captured data was analyzed, excluding the DNS tunneling attack. The frequency of the top 10 domain names is shown in Table 6.1. We found that *encipher.io* domain name is the value that occurs most frequently in the data set, having 344776 DNS requests for the three-month window period while the *youtube.com* domain name has the lowest frequency number in that list, having 68956 DNS requests.

Table 4.2: Frequency of Top 10 Domain Names

Frequency	Domain Name
344776	encipher.io
339618	antsle.com
168649	us-east-1.amazonaws.com
148326	google.com
143410	wd2go.com
132706	app-measurement.com
85932	googleapis.com
80354	Microsoft.com
76220	facebook.com
68956	youtube.com

We also analyzed the complete dataset to determine some dataset relationships between normal and malicious DNS requests. The Analyzer module marks the dataset with two types of labels: *No_attack* and *DNS_ATTACK*, which helps the machine learning algorithms train on this dataset with the labels and then tests them while excluding the labels. Fig 4.4 represents the malicious DNS requests against the normal one. Our dataset has 3646 regular DNS requests; on the other hand, the dataset has 33 DNS malicious requests that been generated using various DNS tunneling tools, as we mentioned earlier.

Even though we have only 33 DNS malicious requests, some of them were found in the top 25 list, and that is because these tools generate a large number of requests per minute.

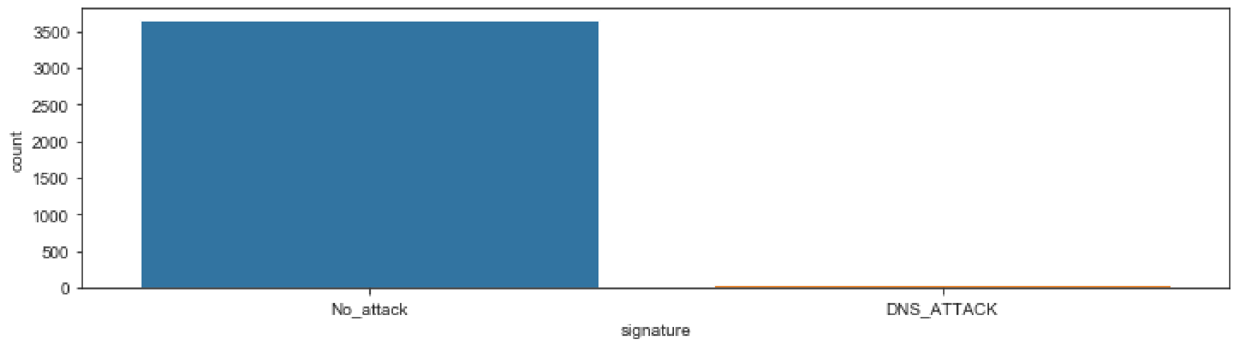


Figure 4.4: The Normal and the Malicious DNS requests in the dataset

Thus, the number of DNS requests for two hours of performing DNS tunneling attack will be higher than the three-month window period of capturing DNS requests.

The DNS Tunneling attack encapsulates the TCP protocol packets and sends them over the domain name system. Due to the DNS protocol limitations, the TCP packets need to be split into pieces to be sent with the DNS protocol. As a result, a successful attack sends many DNS requests per minute in order to perform a certain task. Thus, to confirm that we examine and study the DNS features of the normal and the malicious DNS requests within our lab. Fig 4.5 shows the distribution of the overall total requests for the normal and the malicious DNS requests for the dataset. This shows that the total number of DNS requests with the *DNS_ATTACK* label is much larger compared to DNS requests labeled *No_attack* and also have very different distributions. The different distributions demonstrate graphically why the machine learning algorithms are able to distinguish between normal DNS requests and malicious DNS requests.

We also observed the total number of the TTL average and the data length features in both the normal and the malicious DNS requests. Fig 4.6-A shows the average number of TTL in the DNS requests labeled as *DNS_ATTACK* compared to the normal DNS requests. We observed that the average number of the TTL in the malicious DNS requests is consistent because the DNS tunneling tools use a fixed number of TTL values to send and receive DNS requests. The average total number of TTL in DNS requests that labeled as *No_attack* vary. Similarly, we examined the average data length number between both the normal and the malicious DNS requests. Fig 4.6-B shows that the average numbers in data length in both are varied, and the main reason is that some of the DNS tunneling tools using the DNS data feature to send and receive data during the attack. The distributions of normal DNS traffic

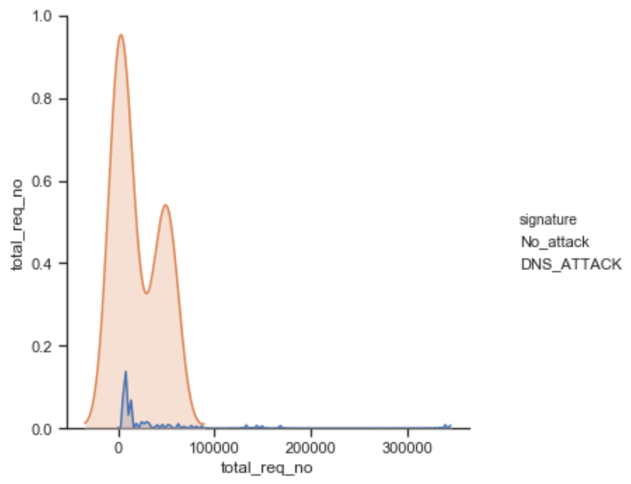


Figure 4.5: The Total Number of the Normal and the Malicious DNS requests in the dataset

versus the DNS are very different.

4.2.4 Evaluation Metrics

In this section, the evaluation metrics for evaluating the detection system are defined. Accuracy and effectiveness are essential factors in the evaluation process, which describes the ability of the detection system to differentiate between intrusive and non-intrusive activities.

Confusion Matrix

The confusion matrix is a tool for determining the performance of the detection system module. It contains information about actual and predicted classifications. Most researchers primarily focus on evaluating and measuring the intrusion detection systems (IDSs) based on the accuracy and effectiveness in terms of false alarm rate and the percentage of attacks that are successfully detected [51]. Similarly, this work will evaluate and measure the proposed system using the confusion matrix as defined below:

1. True positives (TP) - the number of intrusive activities that are correctly detected by the detection system.

2. False positives (FP) - the number of normal or non-intrusive activities that are incorrectly identified as an attack by the detection system.

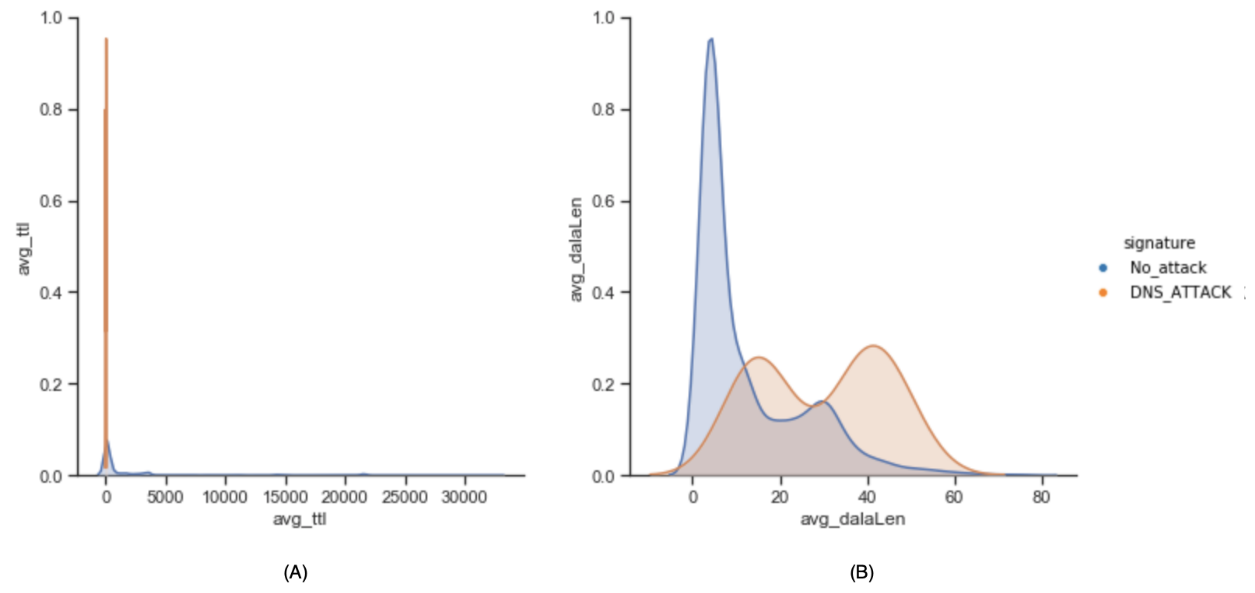
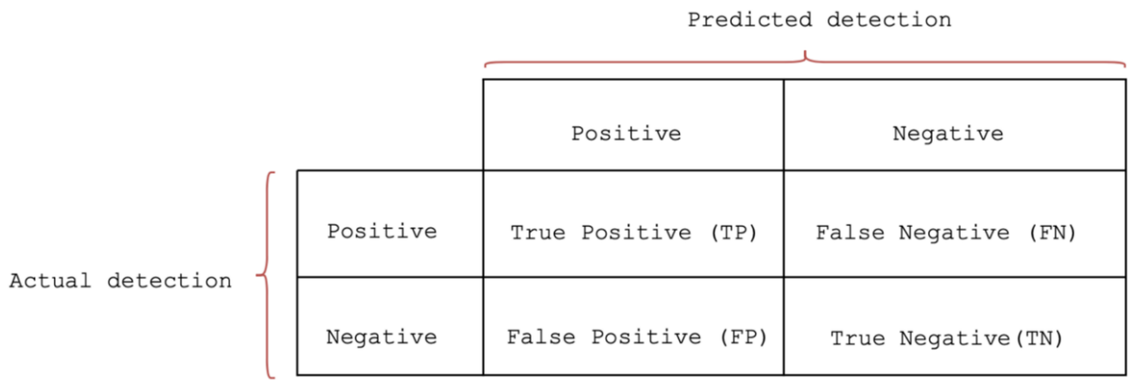


Figure 4.6: The Total Number of the TTL and Data Length in the Normal and the Malicious DNS requests in the dataset

3. True negatives (TN) - the number of normal or non-intrusive activities that are correctly identified as normal by the detection system.

4. False negatives (FN) - the number of intrusive activities that are incorrectly marked as normal or non-intrusive by the detection system.

Figure 4.7 provides information about the confusion matrix of TP, TN, FP, and FN, which represents true and false classification results.



		Predicted detection	
		Positive	Negative
Actual detection	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative(TN)

Figure 4.7: The Confusion Matrix

Metrics from confusion matrix

In this section, different classification measures that are derived from the confusion matrix values are described. The confusion matrix metrics produce numeric values that are simple to compute and compare [51], including true-positive rate (TPR), false-positive rate (FPR), precision (PR), and accuracy, using the following formulas [14]:

1. True-positive rate (TPR) is a ratio between the number of true positives and the sum of the true positives and false negatives. It is used to measure the proportion of actual positives cases that are correctly identified as a DNS tunneling attack. Also, recall and sensitivity are other machine learning terms that refer to a true-positive rate.

$$TPR = \frac{TP}{TP + FN} \tag{4.1}$$

2. False-positive rate (FPR) defines as the proportion of negative cases that are incorrectly identified as positive cases in the DNS traffic.

$$FPR = \frac{FP}{FP + TN} \tag{4.2}$$

3. Precision (PR) is the ratio of correctly predicted positive attacks to the total predicted positive attacks. For example, consider an email spam detection system. A false positive means that a non-spam email (actual negative) has been identified as spam (predicted spam). The email user might lose important emails if the precision is not high for the spam detection model.

$$PR = \frac{TP}{TP + FP} \tag{4.3}$$

4. Accuracy is the ratio of correctly predicted attacks, true positives plus true negatives, to the sum of true positives, false positives, true negative, and false negatives. If the system performance has a high accuracy value, then the system is considered as better than a system with lower accuracy.

$$Accuracy = \frac{TP + TN}{TP + +FP + TN + FN} \tag{4.4}$$

5. F1-Score defines as the harmonic mean of precision and recall models, and it is a method of combining and weighted average of both precision and recall.

$$F1 = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{4.5}$$

4.2.5 Results

In this section, the results of the work are discussed. The aim of this research designs a detection method that identifies and detects the DNS tunneling attacks from network traffic. Our method starts by capturing the DNS traffic of the home-base network for a three-month window period using the Parser module. The dataset has more than 3,500,000 DNS requests before sending it to the Analyzer module. Then, the Analyzer module processed the DNS requests and preprocessed the requests to filter the dataset to be ready for the machine learning algorithms. The dataset is being used during the experiment. It contains normal DNS requests and malicious DNS requests, with 3646 normal DNS requests and 33 malicious DNS requests that were generated using various DNS tunneling tools. Finally, the Detector module splits the dataset into the training and testing dataset and then applies machine learning algorithms to identify and detect any malicious activities within the DNS traffic. The results shows that all of the machine learning algorithms that were used in the experiment achieve 99% accuracy to identify the attack. Table 4.4 compares the machine learning classifiers, including Decision Tree, Logistic Regression, Naïve Bayes, K-Nearest Neighbors, Random Forest, and Support Vector Machine classifiers based on the precision, recall, F1-score, and accuracy.

Table 4.3 shows the confusion matrix. The machine learning classifiers were trained on 50% of the dataset, while the other part, 50% of the dataset, was used to test result on these classifiers. During the testing phase, the dataset included a subset of the dataset, which was selected by the classifier. The subset of the dataset was 1840 domain names, including 1823 non-malicious domain names and 17 malicious domain names. Four of the machine learning classifiers, Decision Tree, Logistic Regression, K-Nearest Neighbors, and Random Forest classifiers, achieved identical results on the dataset in terms of the values of precision, recall, and the F1-score. The results show that the true-positive value of these classifiers was able to identify the 17 cases of the malicious domain names as a DNS attack. At the same time, there were no cases identified as false-negative value, and 1820 cases identified as normal domain names and labeled them as *No_Attack*. On the other hand, the result shows that we have three miss cases, false-positives, where the classifiers labeled the malicious domain name as *No_Attack*. Even though Naïve Bayes classifier achieved a high accuracy which 99%, the confusion matrix result shows that there are missed cases in identifying the DNS attacks. Naïve Bayes classifier was able to identify 16 malicious

Table 4.3: The confusion matrix values for the machine learning classifiers.

ML. Classifier	(TP)	(FN)	(FP)	(TN)
NDecision Tree	17	0	3	1820
Logistic Regression	17	0	3	1820
K-Nearest Neighbors	17	0	3	1820
Random Forest	17	0	3	1820
Naïve Bayes	16	1	5	1818
Support Vector Machine	14	3	0	1823

cases out 17. In addition, the results show that the classifier incorrectly labeled one case as *DNS_Attack*. In the case of false-positive, the classifier was incorrectly labeled five cases as a *No_Attack*. Finally, the Support Vector Machine classifier was able to identify 14 cases of the DNS attacks out 17, which means it three two cases. The Support Vector Machine classifier had false-negatives, incorrectly labeling three cases as DNS_Attack. It had no false positive cases, and the true-negative values of shows the classifier identify 1823 cases as normal.

In addition, we performed other experiments of splitting the data into various ratios and obtained slightly different but similar results. We trained the machine learning classifiers on 30% of the dataset while the machine learning classifiers were tested by 70% of the dataset, and we also obtained a 99% accuracy on all machine learning classifiers. The dataset that been used for the experience had 1104 of Normal DNS requests and 11 of DNS attacks. Most of the machine learning classifiers were able to identify all the malicious DNS requests except the Support Vector Machine classifier was unable to identify all the DNS malicious cases where it missed three cases out of 11 malicious DNS requests. In general, using such a split of dataset ratio, the classifiers labeled one case as normal, but it was a malicious request. In contrast, the Naive Bayes and Support Vector Machine Classifiers marked three cases and labeled them as normal while was malicious requests. Next, we also split the dataset into 70% for training purposes and 30% of the dataset for testing purposes, and the dataset had 2552 cases of normal DNS requests and 24 cases of malicious DNS requests. We obtained 99% accuracy as an average of all machine learning classifiers. Most of the machine learning classifiers were able to identify all the malicious DNS requests except the Support Vector Machine classifier was unable to identify all the DNS malicious cases where it missed three cases out of 11 malicious DNS requests. Most of the classifiers missed three cases of malicious DNS requests, while the Naive Bayes missed five cases of malicious DNS requests. The Support Vector Machine classifier did not miss any malicious cases, but instead, it marked 9 cases as malicious DNS requests where were not malicious.

Table 4.4: Compared Results Among Machine Learning Classifiers

ML. Classifier	Precision	Recall	F1-score	Overall Accuracy
Decision Tree	0.85/1.00	1.00/1.00	0.92/1.00	99.836
Logistic Regression	0.85/1.00	1.00/1.00	0.92/1.00	99.782
Naïve Bayes	0.76/1.00	0.94/1.00	0.84/1.00	99.728
K-Nearest Neighbors	0.85/1.00	1.00/1.00	0.92/1.00	99.673
Random Forest	0.85/1.00	1.00/1.00	0.92/1.00	99.836
Support Vector Machine	1.00/1.00	0.82/1.00	0.90/1.00	99.836

Conclusion

To summarize, the machine learning-based detection system is a network-based solution that either passively captures all DNS traffic or reads DNS packets from existing PCAP files to detect the DNS tunneling attack. It is based on the previous work done in [40] which generates a graphical signature text representation that labels the dataset. The detection system consists of three main modules: the Parser, the Analyzer, and the Detector, where the Parser captures and feeds the system with DNS data. The Analyzer module prepares and analyzes the data for the next module which is the Detector. The Detector module works as a decision-maker built based on supervised learning and pre-trained objects, determining whether or not the DNS tunneling attack occurs. During the experiment, we captured and collected three months of DNS traffic within a private network and performed real DNS tunneling attacks within a controlled private network to simulate the attacks. Then we used these captured data and simulated attacks data to train, test, and evaluate our detection system. As a result, the system achieves 99% accuracy to detect all simulated cases of the attacks with a low false rate. In future work, we intend to implement a DNS proxy server that aims to detect such attacks in real-time using the proposed system in this work.

5 System Methodology

In this chapter, the detection system is introduced and discussed the system architecture design and its main components. Our approach is a network-based detection system that can identify and detect malicious DNS requests, including DNS tunneling, data exfiltration, and command and control channels communications over the DNS protocol.

5.1 Background

The Domain Name System (DNS) considers as one of the most significant elements in the Internet infrastructure. Many protocols and applications such as HTTP, email, and FTP applications rely on DNS to resolve human-readable domain names to the corresponding IP address. Due to the DNS's importance, adversaries misuse and exploit the DNS protocol in many various ways to achieve their malicious goals. Also, DNS-based attacks are one of the most serious cyber-threats used to steal an online identity. According to a global 2019 DNS Security survey, 91% of malware uses DNS services to establish their cyberattacks, and a DNS attack had targeted 82% of survey respondents in 2018 [15]. The 2020 global DNS threat report shows that the number of victimized and attacked organizations via DNS-based attacks increased by 82%. Moreover, the report exhibits the top DNS-based attacks, including DNS phishing, DNS-based malware, DNS amplification, and DNS tunneling attacks. DNS attacks and business consequences are explicitly interlinked, and there are direct strong measurable business impacts. The report shows that such attacks targeted various sectors such as healthcare, education, government, transportation, and utilities and cost more than one million dollars as impacts on their businesses. This section introduces the detection system, which is a network-based solution against malicious DNS requests.

5.2 DNS Proxy

The system detection approach is a network-based detection system that works as a DNS proxy server between clients and public DNS servers. A DNS proxy server acts as a regular DNS server that takes DNS queries from network clients and forwards them outside the network into a private or public Internet Domain Name Server. One of DNS proxy's advantages is that it has full control over the incoming and outcoming requests using either whitelist or blacklist domain names for blocking certain domain names. It may also have the ability to cache DNS records for speeding up the domain name lookup process. DNS Proxy server features are available in almost every enterprise firewall appliance devices. In this work, we used the DNS proxy concept to fully control the DNS queries and requests as well as block the attacks as soon as identified and detected. The current state of the detection system is to identify, detect, and block DNS attacks.

5.2.1 DNS Proxy as a Detection System

In this work, we used the term "DNS proxy" to implement the detection system. Our Golang code is hosted on a dedicated server within the network to run as a DNS proxy server to have a full control over the DNS traffic, including the incoming and outcoming DNS queries. Therefore, in order to implement our methodology, there is a particular setup that needs to be set. First, the network router needs to configure the DNS IP address section to point to our DNS proxy server. The DNS proxy is placed within the home network to intercept clients' DNS queries. As soon as the DNS proxy receives the DNS queries, it applies several analytical functions and then forwards it to the actual DNS server, the Internet provider DNS server, or a public DNS server. After the internet provider DNS server receives an answer, it replays it to the DNS proxy to send it back to the client, as shown in 5.1.

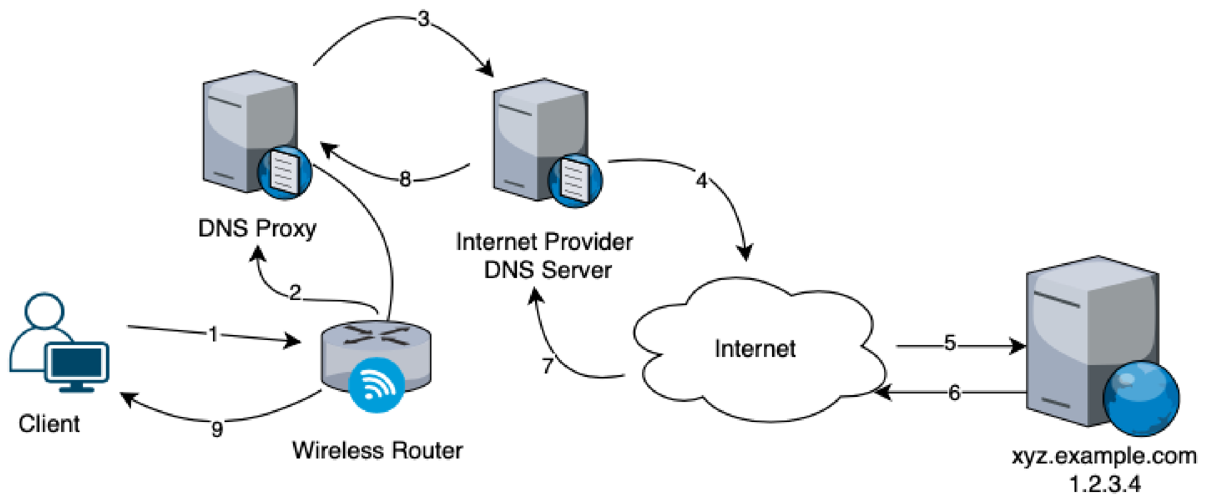


Figure 5.1: DNS Proxy

5.3 System Design

The detection system's main purpose is to distinguish the DNS traffic's abnormal behavior and block malicious domain name requests. The detection system needs to be built with multiple modules using different approaches and communication patterns in order to detect both the DNS tunneling attacks and the DNS exfiltration technique, including malicious software and Malware Bots. For example, in some cases, the DNS exfiltration attack is used as the low-throughput DNS technique. The attacker sends a DNS request every x minutes to exfiltrate sensitive data outside the organization's network through security appliances without been detected. In contrast, the DNS tunneling attacks method must use the high-throughput DNS technique due to the communication patterns. For example, it has to send and receive an enormous amount of data over the DNS protocol to establish a VPN connection.

This section describes the network-based detection system's system architecture design, consisting of two main analysis modules: The Payload Analysis (PA) and the Traffic Analysis (TA) modules. As soon as the DNS proxy receives a DNS query from a client, the first module, the payload analysis, intercepts and operates in real-time. The second module, the traffic analysis, works every x minutes to parse, analyze, and detect the incoming DNS requests, where x is configured based on the daily incoming traffic in the network. We implemented two main analysis modules to detect most malicious DNS requests, including low and high throughput DNS tunneling and exfiltration techniques, whereas identifying and detecting low throughput techniques via payload analysis module in real-time and high throughput techniques via traffic analysis module every x minutes. Fig 5.2 shows the DNS proxy detection system's architecture design, including the payload and traffic analysis modules.

5.3.1 Parsing DNS Data

The detection system works as a DNS proxy, which means it intercepts incoming and outcoming clients' DNS traffic. There are two methods for collecting DNS data to analyze. The detection system feeds the modules with DNS data by either passively captured DNS traffic at the network edge, via DNS proxy server, or read from existing PCAP files via the Parser on the analysis module. When the DNS proxy runs, it analyzes the received DNS queries via the payload analysis module for any malicious activities within the DNS requests

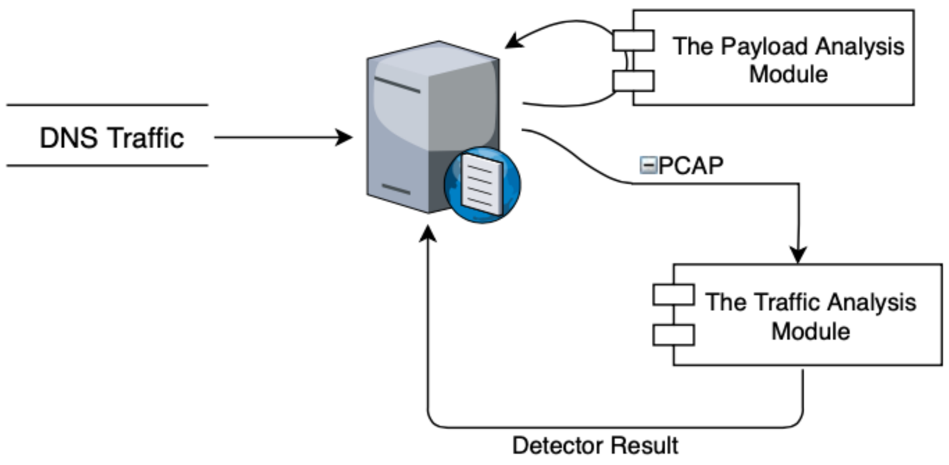


Figure 5.2: The Architecture Design of DNS Proxy Detection System.

and then sends it back to the client. Meanwhile, it also passively captures the DNS requests and stores them into a PCAP file to supply the traffic analysis module with DNS data for further and more extensive analysis.

5.3.2 Feature Extraction

The DNS protocol's nature design is to exchange specific information related to domain names [2]. It was not designed to transfer data over the DNS protocol. The DNS protocol has various limitations, such as the length of any label of the domain name is restricted to between 1 and 63 bytes as well as a full domain name is limited to 255 bytes of letters, digits, hyphens, and separators are included [3]. Even though these limitations exist, adversaries may abuse and leverage the DNS protocol to transfer data to their authoritative name server and establish a covert channel for their activities. The attacker can include as much information in the DNS queries to pass the data. Instead of sending legitimate subdomains, adversaries take advantage of the domain length limitation and encode some data in the domain name.

After parsing the DNS data for the system, the features are extracted to analyze. Note that each module intercepts and examines the DNS traffic based on various features and characteristics. Thus, in this work, it is required to parse and extract features for each module individually. The payload analysis module monitors and analyzes based on features such as length of DNS queries and responses, information entropy, uncommon record types, blacklisted domain names, and statistical analysis. While the traffic analysis monitors the

DNS traffic for other features such as the volume of DNS traffic per IP address, the frequency of DNS traffic per domain, visualization of the traffic, the number of hostname per domain, and other analysis features. Therefore, a set of features needs to be considered.

5.3.3 The Payload Analysis Module

The Payload Analysis (PA) technique focuses on identifying and analyzing critical features of the DNS queries. It intends to focus on monitoring and observing the DNS traffic that uses the particular patterns and indicators. Previous research and studies show that malicious domain names can be found and represented in different forms and encoded characters. This section describes the method of the payload analysis module approach. Implementing the payload analysis module is to identify and detect the low-throughput data exfiltration or DNS tunneling. One technique to bypass the firewall and in-place security appliances is to use low-throughput techniques. The adversary sends a malicious request every minute to avoid being detected. Adversaries use various data encoding techniques and communication patterns in the DNS protocol to carries binary files, sensitive or stolen data outside the organization's network. Examples of data encoding techniques include *MD4*, *MD5*, *Base32*, *Base64*, *hex*, and others. Thus, this section exhibits the approach used to analyze and identify the DNS protocol for these techniques.

The Payload analysis module relies on DNS features, focusing on collecting and analyzing DNS query details, including statistical analysis and the information entropy. Thus, we need to perform DNS feature extraction and examine them to detect if any DNS malicious occurs. In the payload analysis module, the feature extraction phase works in real-time on the network edge side, using a DNS proxy. Every time the DNS proxy server receives a DNS request from a client, it extracts the DNS features for the analysis process. It then passes it to the analysis module, which applies statistical and analytical functions to captured DNS requests for analyzing to determine malicious activities. DNS features extraction includes the Fully Qualified Domain Name (FQDN), query record type, a subdomain of FQDN, and the subdomain's query length, encoding the query data's subdomain, blacklisted and whitelisted domain names, and information entropy.

1. **Uncommon record types**: DNS Tunneling tools rely on certain types of DNS queries due to the limitation of the DNS protocol [3]. As we mentioned before, the DNS protocol is not designed for transferring data but instead is used to exchange information

about the domain name. Thus, the domain name system is limited to a certain number of bytes of letters and symbols. However, that depends on the type of DNS query that is being used. The DNS text (TXT) type record lets system administrators describe the service of the domain name by entering human-readable notes. Therefore, many DNS tunneling tools leverage the DNS *TXT* type record to transfer as much data as possible. In this work, the detection system monitor uncommonly used record types such as *TXT* and *NULL*.

2. **Data Encoding**: Adversaries take advantage of the domain name system to pass binary files, sensitive information, or stolen data over the DNS protocol. They most likely include encoded or encrypted data in the domain name before passing it to make it look like a part of the domain name request. For example, if the *thepassw0rd.example.org* domain is queried, the adversary's authoritative name server, for *example.org*, receives the *passw0rd* as a string of hostname but interpret it as incoming data instead. Also, adversaries may use some other encoding techniques to transfer text or binary data to their servers. Another section that may have encoded or encrypted data is the description section in the *TXT* requests. Therefore, the detection system monitors the domain name system and looks for encoded or encrypted data, including *Base32*, *Base64*, *Based128*, *hex*, *md5*, *md4*, and other Non-English characters.

3. **Length of DNS query**: The DNS tunneling tools utilize the domain names to transfer as much data as possible over the DNS protocol. Therefore, the domain names' and hostnames' lengths are considered abnormal behavior if they exceed a certain length value, DNS requests' lengths, including the length of characters and labels, the number of dots. In a 2012 presentation at the RSA conference, Ed Skoudis identified a new DNS-based Command and Control of malware that uses the DNS exfiltration technique to transfer data over the DNS protocol [52, 53]. Based on that finding, they recommend most of the Fully Qualified Domain Names (FQDN) longer than 52 characters and most hostnames that are longer than 27 unique characters should be considered suspicious [52, 53].

4. **Blacklisted and Whitelisted domain names**: These lists are used to improve the detection system's results over time. We are filtering the capture DNS data using blacklisted domain names to detect the malicious domain name before processing it

for faster performance. Using the whitelisted domain names is to reduce the false-positives rate of the results over time. As we show in the results chapter, we found many use cases of legitimate domain names are using the DNS exfiltration to establish a connecting channel between clients and servers to transfer data over the DNS protocol.

5. **Entropy of hostnames**: According to the DNS protocol RFC in [54], the valid DNS requests must consist of English letters, digits, and hyphens. Therefore, the legitimate domain name requests most probably could be found in the English word dictionary while the DNS requests that have encoded names could have higher entropy than the English words due to extensive characters [9]. In the article found in [55], the author recommended using name entropy as one factor in detecting the DNS tunneling attacks as the legitimate domain name and hostnames are using the English language that has redundancy. In contrast, the encoded data of hostnames are long and have higher entropy.

6. **Statistical analysis**: Another approach to detecting the DNS tunneling attacks is by applying statistical functions on the captured DNS data. DNS tunneling technique queries unique requests every time sending data to avoid getting cached by the browser or the DNS server. Therefore, we applied statistical functions on DNS data to observe the frequency of the specific characters' occurrence. In this work, we also applied statistical functions for both the payload and the traffic analysis modules, such as the total length of FQDN and hostnames, the total number of labels in the FQDN, the volume of DNS traffic per IP address, and the number of hostnames per domain. Also, we find the frequency of DNS traffic per domain, hostname, source, and destination IP addresses, TTL.

Figure 5.3 provides an overview of how the process of the payload module works, if a client tries to browse *xyz.example.com* website. As soon as the DNS proxy receives the DNS request, it extracts the feature of the DNS request for real-time analysis. It gathers and analyzes some features, including query record type, query's length, subdomain name. Then, it passes it to the analysis module, which looks for abnormal behaviors such as information entropy, blacklisted domain name, malicious query record type and length, abnormal characters, or encoding data types within the subdomain name. Finally, it tags the DNS requests as malicious and returns the result status of the DNS request to the DNS proxy to process it.

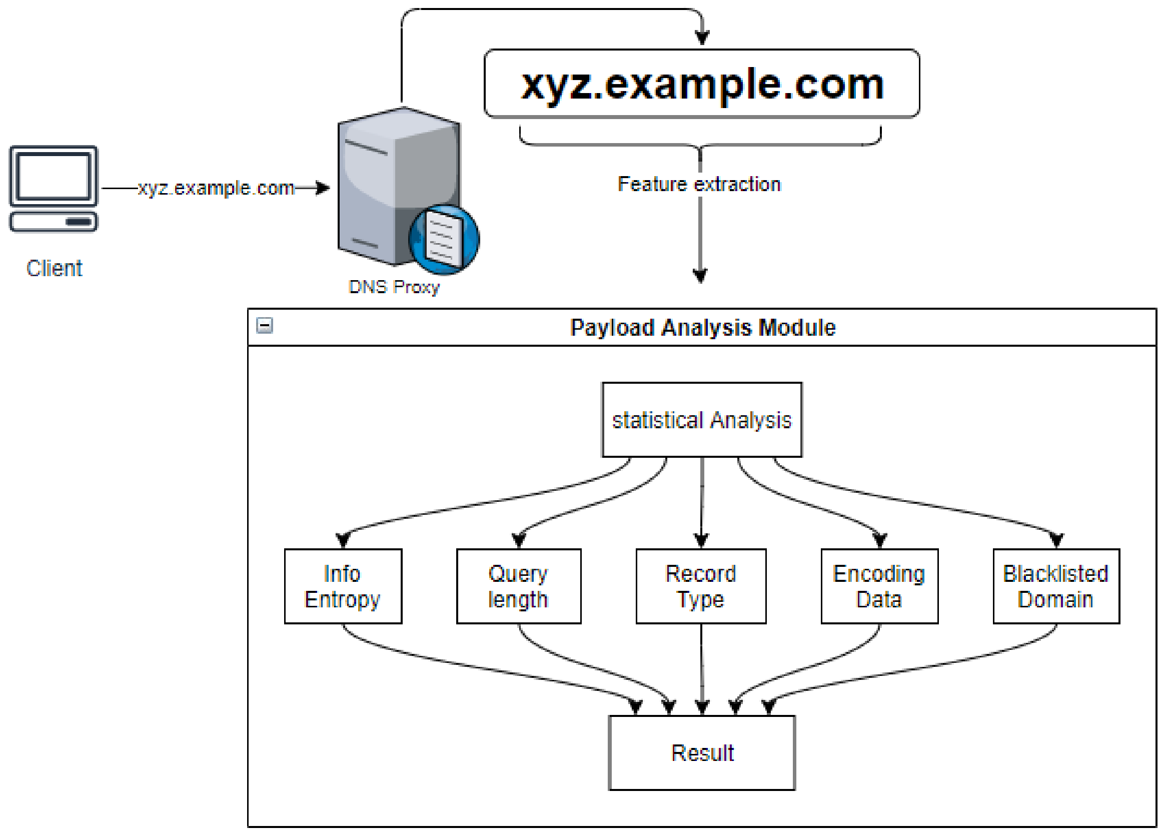


Figure 5.3: The Real-time Payload Analysis Module.

5.3.4 The Traffic Analysis Module

The traffic analysis module distinguishes and detects DNS-based malicious activities on gathered DNS traffic. In this module, the detection system parses DNS traffic every x minutes and analyzes it based on a visualization technique and machine learning. The detection system works based on the DNS proxy concept, which controls the network's DNS queries and responses. In this module, we utilized the detection system module in [40] that applies the parallel coordinates technique to identify and detect the DNS tunneling attacks. The DNS proxy captures and stores the DNS traffic every x minutes. Then, it passes the captured *PCAP* file to the traffic analysis module to process for further analysis.

This module detects the DNS-based abnormal behaviors where the DNS tunneling tools are used to establish covert communication channels or VPN over the DNS protocol. This technique sends a large number of DNS requests per second to achieve establishment, called the high-throughput DNS tunneling. In addition, this module has the ability to identify modern attacks and tools. In some cases, adversaries use modern methods to bypass the internal system's security. For example, the *PacketWhisper* tool uses legit domain names in the attack, making it more difficult to detect and exfiltrate sensitive data [56]. We show more details about the results against *PacketWhisper* in the Results chapter.

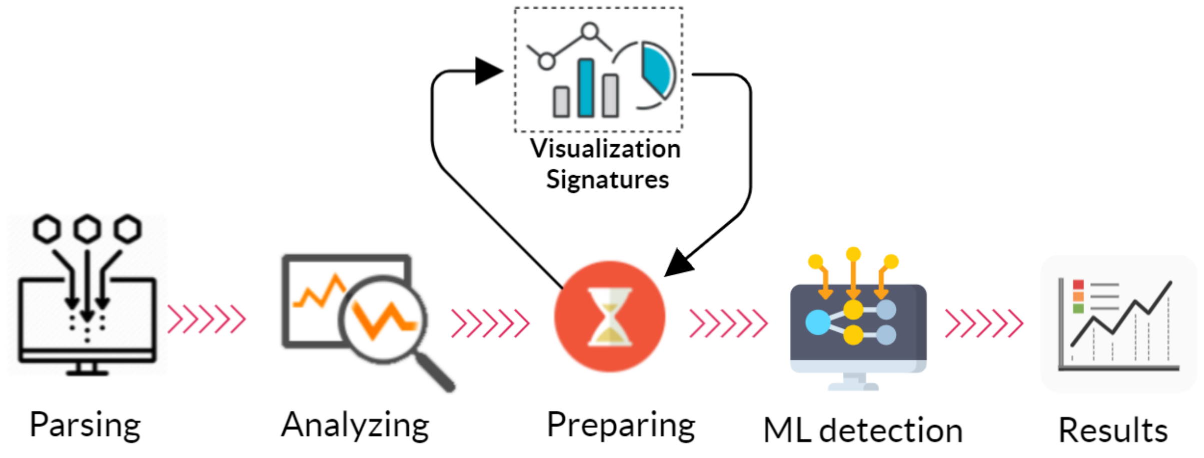
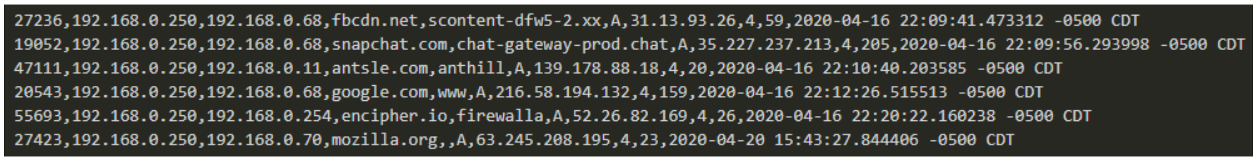


Figure 5.4: The Overview of Traffic Analysis Module.

Figure 5.4 highlights the overview of the traffic analysis module phases. At the moment the PCAP file ready, the traffic analysis module starts the process of analyzing. First, the Parser is parsing the PCAP file and gathering the DNS required information for the next stage. The DNS packet has much information; however, the traffic analysis module parses and collects the following information: ID, source and destination of IP address, domain name, subdomain name, Service IP (SRV) address, DNS record, data length, TTL, date, and time. Figure 5.5 shows an example of captured DNS data during the parsing phase in the traffic analysis module.



```
27236,192.168.0.250,192.168.0.68,fbcdn.net,scontent-dfw5-2.xx,A,31.13.93.26,4,59,2020-04-16 22:09:41.473312 -0500 CDT
19052,192.168.0.250,192.168.0.68,snapchat.com,chat-gateway-prod.chat,A,35.227.237.213,4,205,2020-04-16 22:09:56.293998 -0500 CDT
47111,192.168.0.250,192.168.0.11,antsle.com,anthill,A,139.178.88.18,4,20,2020-04-16 22:10:40.203585 -0500 CDT
20543,192.168.0.250,192.168.0.68,google.com,www,A,216.58.194.132,4,159,2020-04-16 22:12:26.515513 -0500 CDT
55693,192.168.0.250,192.168.0.254,encipher.io,firewalla,A,52.26.82.169,4,26,2020-04-16 22:20:22.160238 -0500 CDT
27423,192.168.0.250,192.168.0.70,mozilla.org,,A,63.245.208.195,4,23,2020-04-20 15:43:27.844406 -0500 CDT
```

Figure 5.5: Example of Parsed DNS Data

Next, the analysis phase applies statistical functions on the captured DNS data for each domain name individually, such as the average number of data length, TTL, subdo-

Table 5.1: DNS Tunneling Visualization Signatures

Attack	Signature	Representation	Normal\Malicious
Traditional D. T.	1,1,1,m,1,1	Attack_Type1	True
Botnet D. T.	m,1,1,m,1,1	Attack_Type2	True
Heyoka D. T.	1,m,1,m,1,1	Attack_Type3	True
Heyoka /Botnet D. T.	m,m,1,m,1,1	Attack_Type4	True
No Attack	Not Above	0	False

main's length, the total number of domain requests, source and destination IP address, service IP address, and unique hostname requests per domain name. We also used the virtualization module using the parallel coordinates technique in section x to generate relation signatures that determine the DNS tunneling attack occurs or not. Table 5.1 show more details about all kinds of attacks and their signatures. In the preparation phase, these signatures were used to label the analyzed DNS data for a machine learning classifier for training purposes and test and evaluate the machine learning classifier results. Also, the Alexa and Cisco Umbrella top one million domains [49] are used to identify and label the captured domain name as legitimate.

The visualization of the traditional DNS tunneling attack shown in Figure 1.4, plots the quantity of source IP addresses, destination IP addresses, domain names, subdomain names, time-to-live (TTL) values, and time values. An easily identifiable pattern is converted to a signature represented as the tuple *(1, 1, 1, m, 1, 1)*. That means there is one source IP address, one destination IP address, one domain name, multiple (m) subdomains, a single TTL, and a single or small window of time. The traditional DNS tunneling attack, Botnet DNS tunneling attack, Heyoka DNS tunneling attack, and Heyoka/Botnet DNS tunneling attack all have unique visualizations converted to attack relationship signatures. Finally, the machine learning phase reads the prepared data to process it and provides detection results for each domain name. In Chapter 4, we provide more further details about the machine learning classifier results where we experimented with testing and evaluating this module separately.

5.3.5 System Features

In this section, we discuss some of the detection system features. The detection system is a network-based solution that identifies, detects, and blocks DNS-based attacks, including DNS tunneling attacks and DNS exfiltration. This work aims to determine the DNS-based attacks as soon as it happens in different forms, including the low-throughput and high-throughput DNS tunneling attacks. We implemented the detection system to work in two main modes: real-time and offline modes for better performance. The detection system is coded using Golang programming language for fast and stable performance. The detection system runs based on the Linux operating system and acts as a DNS proxy within the network to fully control the DNS queries and responses. Thus, some traditional DNS servers utilize blacklisted, and whitelisted domain names for blocking advertisements popup windows, such as *Pi-hole* [57]. In this work, the detection system can also use the concept of using blacklisted and whitelisted to improve the overall performance. The detection system also has the ability to perform other services that the traditional DNS proxy provides, such as caching domain names, which can cache DNS records for speeding up the domain name lookup process.

Also, in this work, the detection system blocks malicious DNS requests after being detected in real-time by caching and pointing the malicious domain name's IP address to *0.0.0.0.* in the DNS proxy. Thus, every time the malicious domain name is requested it replies with the loopback address. We also implemented an API web interface for the detection system that provides manual operations such as creating, removing, and modifying domain name records within the cache system. This feature also will be used to design a front-end web interface for future work.

In this chapter's conclusion, we discussed the system architecture of the detection system against DNS-based attacks. The detection system is a network-based solution, as a DNS proxy, that targeting DNS tunneling attacks and DNS exfiltration techniques. Even though the adversary uses the low-throughput DNS tunneling technique, the detection system is able to detect and identify the attack. The detection system consists of two main modules, including the Payload and traffic analysis modules. Thus, the detection system monitors and examines the DNS traffic in real-time employing the payload analysis module. It also performs offline analysis every x minute(s) using the traffic analysis module. We show more details about the detection system results and evaluation in the next chapter.

6 Results and Discussion

In this chapter, we show and discuss the detection system results. The detection system aims to detect and block malicious DNS-based attacks within the network. We employ two detection modes in implementing the detection system that targets DNS tunneling attacks and DNS exfiltration. The real-time mode detects and analyzes the DNS queries from clients, while the offline mode captures and analyzes DNS data for a certain window period. Also, we compared the detection system results with *Snort*, the baseline intrusion detection system. *Snort* is a real-time intrusion detection system, and it is capable of not only monitoring and analyzing the DNS protocol but also other protocols against various Internet attacks. The approach that is used in this work is a network-based solution against DNS-based attacks, including the high and low throughput DNS tunneling attacks.

The network-based solution, which runs as a DNS proxy, consists of three main models: visualization, detection using machine learning classifiers, and real-time modules. Thus, we performed three evaluation experiments individually for each module. DNS tunneling attacks are widely used by either adversaries or malware applications. This type of attack takes advantage of the DNS protocol, which is rarely monitored and accessible by most networks, to bypass firewall and modern security in-line appliances. For example, during our testing and evaluating experiment, a *Firewalla* was employed within the network. A *Firewalla* is a modern security router to detect and block various Internet attacks. While we were performing the actual attacks against the detection system, *Firewalla* could not identify any of these attacks because it does not have the capability of monitoring the DNS protocol.

The proposed detection system is implemented by using visualization techniques and machine learning classifiers to distinguish the DNS-based attacks. Designing a detection system using visualization and machine learning is not new. There are several studies on visualizing network traffic. In addition, there are numerous studies on detecting most Internet attacks. However, there are far fewer studies on DNS tunneling attacks and none of the known techniques perform as well as the technique that is proposed in this work. Thus, this section shows the experimental results for the detection system that we implemented and compares it with others.

6.1 Evaluation

The main focus is on building a network-based detection system for DNS tunneling attacks and DNS exfiltration techniques. To build a robust detection system, we created a local testbed environment to evaluate the system. To benchmark the performance of the detection system, we tested both well-known and custom DNS tunneling tools against the detection system. There are five main tools we used in this work. The five DNS tunneling tools that were selected and used to simulate the attacks include *iodine*, *dns2tcp*, *PacketWhisper*, *DNSExfiltrator*, data exfiltration toolkit (*DET*), and *Cobalt Strike*. The *idoine* and *dns2tcp* tools were used in the experiments to simulate the DNS tunneling attack where the adversaries establish a covert channel over the DNS protocol, such as VPN connection. *Cobalt Strike* is a stealthy advanced persistent threat (*APT*) cyberattack that allows an adversary to gain unauthorized access to a network and remain undetected for an extended period of time. It is also defined as software for adversary simulations and red team operations for performing security assessments, while the *DNSExfiltrator* and *DET* are tools for generating custom DNS exfiltration traffic that simulates malware scenarios. *PacketWhisper* is a modern attack that uses the DNS protocol. *PacketWhisper* is a special case scenario where the attacker is internal and performs the attack from in-site.

This work's main benefit is to recognize and predict new threats with a low false-positive rate and low latency. We calculate the performance evaluation based on three metrics factors: Detection Rate (True-Positive rate), Precision (PR), and Accuracy (ACC). Then, we compared the evaluation results with *Snort*, the open-source detection system. We consider a system that has less false-negatives rate with high accuracy as better.

6.1.1 Experimental Setup

This section discusses the experimental setup for the DNS tunneling detection system. To evaluate the performance of the methodology, we employ the detection system within a local private network. The network environment is a home-based lab consisting of more than 20 devices, including IoTs, servers, mobiles, PCs, and laptops. The detection system is a network-based solution that runs as a DNS proxy to control and intercept the network traffic. The detection system acts as a primary DNS server for all clients in the network to guarantee that we intercept overall clients' requests. The DNS proxy is capable of caching DNS requests for faster query lookup. For this experiment, we run the detection system for four weeks to gather as much data for evaluating the system as well as we performed the DNS attacks using the tools to simulate the attacks.

A Debian Linux server is deployed and hosted in the cloud to act as a public DNS server, and all network clients are using the IP address of the DNS proxy as the primary DNS server. The detection system uses blacklisted domain name lists which protects clients from Internet attacks, including Phishing and spear phishing attacks. As we show in the results section, we blocked many advertisement websites based on blacklisted lists which been created by people.

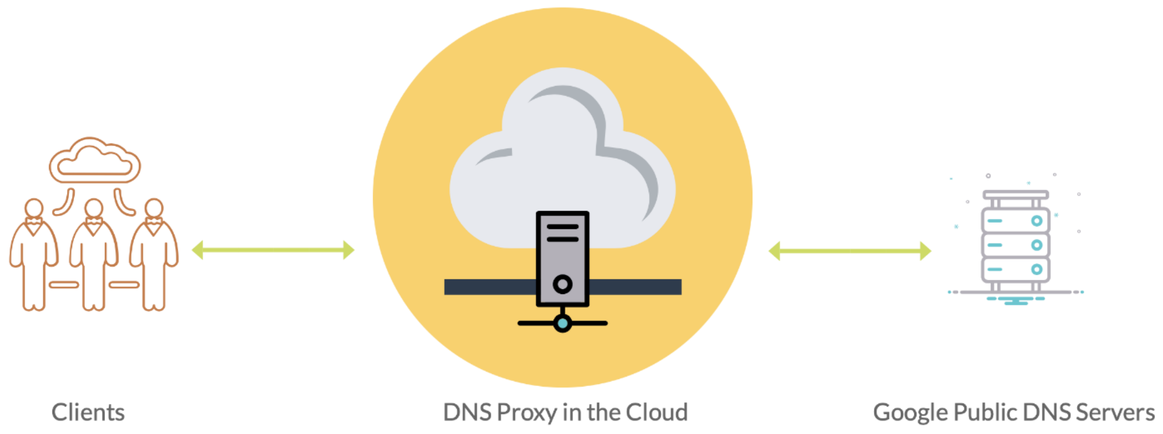
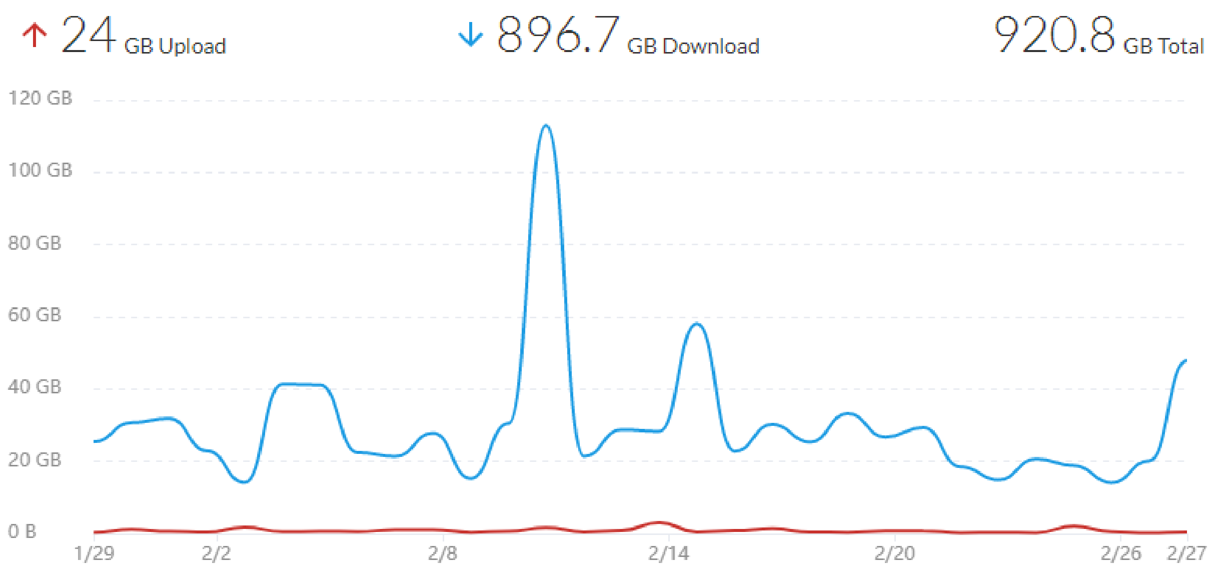


Figure 6.1: An Overview of the DNS proxy Setup.

Figure 6.1 exhibits an overview of the experimental setup of the DNS proxy in the cloud where clients use the DNS proxy as their primary DNS server to resolve DNS queries. Then, as soon as the DNS proxy receives the DNS queries, it checks for exists cached domain names to respond back to the client. If it does not exist, then it forwards the query to Google public DNS servers to resolve the request. Finally, Google public DNS sends back the results to the DNS proxy server, which sends the result to clients.

6.1.2 The Collected DNS Data

In this section, we discuss and describe the captured DNS data gathered during this work. The network environment is a home-based lab that has more than 20 clients. The monthly average network data usage is approximately 25 GB of Internet traffic, an active and busy network. Figure 6.2 gives a clear insight into the network connectivity and data transfer during the evaluation and testing of the DNS proxy server. This work focuses on monitoring and analyzing the DNS traffic and identify malicious DNS-based attacks, including DNS tunneling attacks and DNS exfiltration. Thus, the DNS proxy server was successfully deployed without any issue during the testing and evaluating period.



85

Figure 6.2: Internet Data Transfer during the Four Weeks.

In this work, we used the DNS proxy server and targeted not only the DNS tunneling attacks but also the advertisement websites that gather visitor information and may harm clients. Once the DNS proxy server receives a DNS request, it looks up the domain name in the blacklist. If it exists, then the DNS proxy server blocks the DNS requests. In addition, the DNS proxy applies statistical analysis on captured DNS data for a certain date including information such as total queries, total queries blocked, top ten domain names, top ten blocked domain names, and top clients.

Figure 6.3 focuses and provides information about DNS queries for the four weeks of the experiment's detection system. Note that the number of DNS queries varies between 5000 to 60000 requests, where the total DNS queries is 1,148,018 for the four weeks of the evaluation. The detection system was capable of blocking approximately 20% of the total number of DNS requests based on using the blacklists. These statistics exclude the DNS tunneling attack requests that we performed to simulate the attacks against the detection system. In addition, Figure 6.4.B shows 10% of DNS requests that have been cached in the detection system while the DNS proxy forwards and replies to 70% of the total DNS requests.

Table 6.1: Frequency of Top 10 Domain Names

Frequency	Domain Name
344776	encipher.io
339618	antsle.com
168649	us-east-1.amazonaws.com
148326	google.com
143410	wd2go.com
132706	app-measurement.com
85932	googleapis.com
80354	Microsoft.com
76220	facebook.com
68956	youtube.com

We also analyzed the DNS query types received by the DNS proxy server for the month of testing and evaluation. Figure 6.4.A shows the top five DNS query types including: *A*, *AAAA*, *PTR*, *TXT*, and *CNAME* where the majority of the DNS type is *A* (IPv4), while the *TXT* received by less than 1% of the total DNS query types.

The captured data was analyzed, excluding the DNS tunneling attack. The frequency of the top ten domain names is shown in Table 6.1. It showed that the *encipher.io* domain name is the value that occurs most frequently, having 344776 DNS requests for the four weeks of testing and evalulation while the *youtube.com* domain name has the lowest frequency number in that list, having 68956 DNS requests.

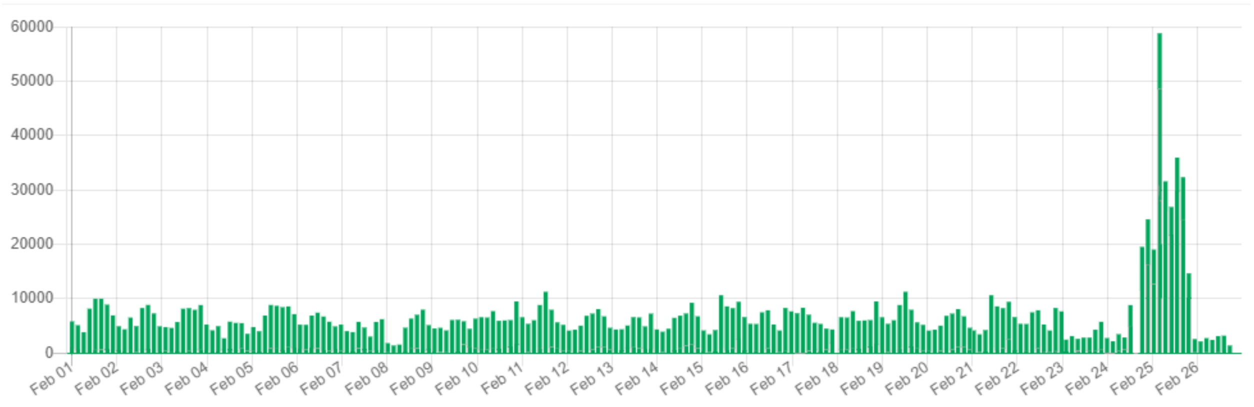
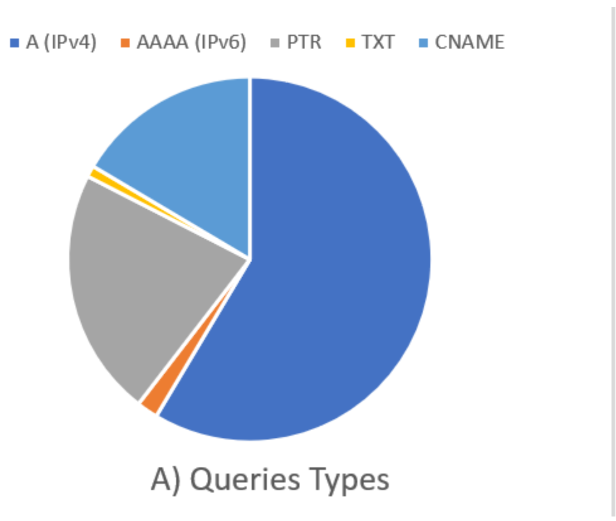


Figure 6.3: DNS Queries over Monthly Time Period.



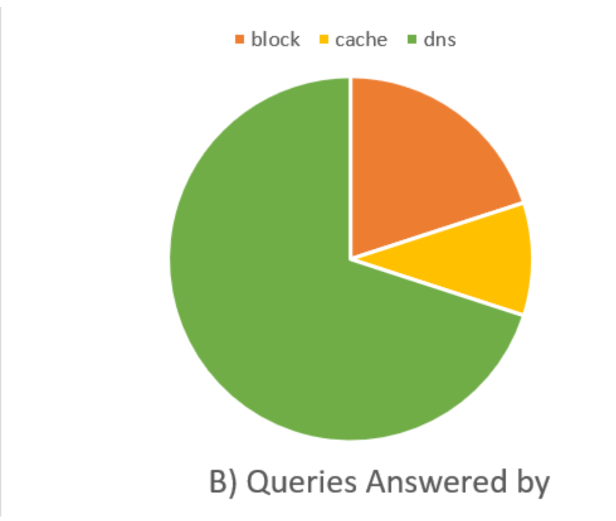


Figure 6.4: Statistics of Queries Types and Answered by.

6.2 Threshold Value

The detection system consists of two detection modules: the payload and traffic analysis modules. Each module uses a threshold value to determine the abnormal activity. For example, we set the hostname's threshold value of the full qualified domain name's length to be 30 characters long in the payload analysis module. This value is selected based on the recommendations in previous work [1,2].

However, it is unclear what threshold value for the number of DNS requests to use within a time frame window to identify malicious domain names for the traffic analysis module. A threshold value is used in the traffic analysis module for machine learning training purposes. After capturing the DNS data, we generate text representation signatures from the visualization. If it is a malicious signature for a domain, we compare the number of DNS requests of the domain name to the threshold value and then label the training dataset as either malicious or not malicious base on the threshold. Thus, there is a need to conduct hands-on experiments for selecting the most beneficial values for numbers of DNS requests within the set period. The traffic analysis module aims specifically to identify and detect high-throughput DNS tunneling attacks over time using visualization signatures and a machine learning classifier. This experiment is significant for the traffic analysis module and its threshold value because it processes and analysis DNS data every x minutes.

This section presents an evaluation experiment to determine the best threshold value for the detection system module, the traffic analysis module. Based on what we observed in the experimental results, there is an obvious way to select an accurate value for the detection system's threshold based on the binary file size and time. It is selected based on an empirical experiment using DNS tunneling attack tools.

The DNS tunneling attack such as TCP over the DNS protocol is usually noisy and sends a large number of DNS requests to establish communication channels. Therefore, for these types of attacks the threshold can be set to be when the amount of DNS traffic is one or two standard deviations above a moving average. Instead, we focused on the more difficult problem of determining the threshold value for the DNS exfiltration technique scenarios. The main reason is this type of technique is to send a specific number of requests depending on the file size.

The tool *DNSExfiltrator* is used to simulate the exfiltration attack that transfers files outside of a compromised network for empirically determining the threshold. Multiple files varying in size from 1KB to 1MB were transferred outside of the network using the *DNSExfiltrator* tool and the number of chunks representing DNS requests were measured. The maximum full domain of 255 bytes and maximum label length of 63 bytes [3] were used in this experiment. If the DNS request and label sizes were reduced to lower values, it would cause even more DNS requests to be sent for a specific file size.

The threshold value is set based on the file size and number of DNS chunks in Table 6.2. We consider the time and the size as significant factors in this evaluation since the training phase of traffic analysis module relies on capturing DNS data every x minutes where setting x is based on the daily network traffic. For example, let's assume that the adversary wants to export a 100KB text file of stolen credit card information outside the organization's network using the DNS exfiltration technique. Table 6.2 shows that the 100KB file needs 600 DNS requests within 25 seconds to successfully exfiltrate it outside the organization network. Then, our detection system needs to capture DNS data every minute, and the value of the threshold needs to be set to less than 600 to detect the attack. In our detection system experiment, we selected that the traffic analysis captures DNS traffic every 30 seconds and set 100 as a threshold value assuming the attacker needs to exfiltrate a 50KB file at least, resulting in approximately 300 DNS requests in 20 seconds. Note that sometimes the adversary could use the delay technique between each DNS request to reduce the noise, but the payload analysis module in this work is designed for these situations.

Table 6.2: Exfiltrating Various Binary files Over the DNS

File Size	Number of chunks	Time	Request size	Label size
1KB	7	1 Sec	255 chars	63 chars
10KB	61	4 sec	255 chars	63 chars
100KB	600	25 sec	255 chars	63 chars
200KB	1199	43 sec	255 chars	63 chars
500KB	2996	1:53 mins	255 chars	63 chars
1MB	6135	4:00 mins	255 chars	63 chars

6.3 Results

This section shows the results of the testing and evaluation process of the detection system. The detection system is a network-based solution to detect and block various DNS-based attacks, including DNS tunneling attacks and DNS exfiltration techniques. In addition, the detection system blocks domain names based on blacklists that are related to advertisement purposes. Most advertisement websites collect sensitive information from visitors, and they also may harm visitors by injecting malicious codes into visitor browsers to execute abnormal activities or redirecting them to malicious websites. The detection system acts as a DNS proxy server capable of fully intercepting DNS traffic within the network to control DNS clients' requests.

In previous chapters, we showed the various modules' results individually, including the Visualization module and the detection based on machine learning. However, this section presents the detection system results as one system where the Payload Analysis module runs in real-time monitoring DNS requests. Simultaneously, the Traffic Analysis module works in offline mode every x minutes based on the statistical functions. By the end of the results section, we also show *Snort* IDS' results, the open-source intrusion detection system, to detect DNS-based attacks and finally compare our detection system's results against *Snort* to determine which detection system is better. The detection system aims to distinguish and prevent DNS tunneling attacks and DNS exfiltration techniques. In order to test and evaluate the detection system against malicious DNS-based requests, we simulate and perform real-world attacks within a private and controlled network environment. In this work, we focus on detecting various DNS-based attack scenarios, including TCP over DNS, DNS exfiltration, and Command and Control (C&C) communication channels.

Table 6.3: Tools to Simulate DNS-Based Attacks

Tool	Simulate Scenario of	Purpose
iodine	TCP over DNS	Establish VPN connection
dns2tcp	TCP over DNS	Establish VPN connection
DNSExfiltrator	DNS exfilitration technique	Exflitrate sensitve data
DET	DNS exfilitration technique	Exflitrate sensitve data
Cobalt Strike	Command and Control	Execute Commands
Cobalt Strike	Command and Control	Upload File
dnscat2	Command and Control	Execute Commands
dnscat2	Command and Control	Upload File
PacketWhisper	Custom DNS attack	Exflitrate sensitve data

Table 6.3 lists the tool, simulated scenario, and the main purpose of the attack. We simulated all of these scenarios against the detection system in the testing and evaluating stage. Note that in the TCP over DNS scenario, the adversary requires sending considerable numbers of DNS queries which makes it noisy. While in the custom DNS exfiltration and command and control scenarios, the adversary sends a limited number of DNS requests, making it less noisy and harder for the detection system to detect it. The *PacketWhisper* tool considers as a modern DNS-based attack that uses legit domain names in the attack, making it more difficult to detect and exfiltrate sensitive data [56].

In the first section to the results, we show the blocking domain names based on advertisement blacklists. During the four weeks of using the detection system, we notice that many companies, such as Google and Microsoft, collect data about visitors. Some of these domain names that are part of gathering data are already in blacklists available publicly. Thus, the detection system was capable of blocking these domain names successfully. Interestingly, Table 6.4 shows the top ten blocked domain names during the experiment and evaluation period. We observed that Android TVs within the private network send requests to *secure-dcr.imrworldwide.com* which is a tracking consumer habit. This Android TV sent 34254 requests which were all blocked.

Next, we performed that attacks where the detection system was in place. The detection system acted as a DNS proxy server so that it could intercept all DNS requests within the private network. We performed various scenarios of DNS-based attacks where

Table 6.4: Frequency of Top 10 Blocked Domain Names

Frequency	Domain Name
34254	secure-dcr.imrworldwide.com
220	mobile-collector.newrelic.com
210	www.googleadservices.com
132	2ecd5.v.fwmrm.net
132	g13v.prd.ads.aws.fwmrm.net
105	cdn.optimizely.com
101	device-api.urbanairship.com
97	device-api.urbanairship.com
94	googleads.g.doubleclick.net
93	sb.scorecardresearch.com

all of the attacks were detected by the detection system successfully. The detection system has two modules: the Payload Analysis (PA) module and the Traffic Analysis (TA) module. In some attack cases, the detection system detected the attack using both the TA and PA, while in other cases it detected the attack using either the TA or the PA.

6.3.1 Results of Experiment 1

This section shows the detection system evaluation results based on the confusion matrix, including the accuracy, missing rates, and other measurements. We installed the detection system in place of the active network during the evaluation, and the total average DNS requests were approximately 1000 per hour. At the same time, we performed 12 actual cases of DNS-based attacks during this experiment, including TCP over DNS, DNS exfiltration, and command and control. The detection system was able to detect all 12 cases successfully with no false negatives value. The detection system identified the attack as soon as received by the DNS proxy server by either the payload analysis or the traffic analysis modules. Still, there are cases where the payload analysis module was not able to detect the attack. The traffic analysis module distinguished the attacks based on the visualization module discussed in Chapter 3.

There is a case, *PacketWhisper* tool, where the detection system has not detected the attack using the payload analysis module but detected it with the traffic analysis module.

Table 6.5: Top 5 Domain Names of False Positives

Domain Name	Company
netgear-07a2d5b3-49d4-9038-f3e9ce19f9ce.2d7dd.cdn.bitdefender.net	BitDefender
e0d67c509fb203858ebcb2fe3f88c2aa.baas.nintendo.com	Nintendo
473fba5a7a73d459537c41fec606664a.fp.measure.office.com	Microsoft
d4d857082fe8f0ff5cc718ae110de2e8.safeframe.googlesyndication.com	Google
4aux00fznmxhbcjiuzyj7hjdjh9jg1614225234.uaid.imrworldwide.com	Nielsen
ic.48ce0c00.0fbf9e.gs2.sonycoment.loris-e.llnwd.net	Sony

PacketWhisper is a combination of DNS queries with text-based steganography. It converts the binary payload file into a list of FQDN. In this attack, there is no need for the adversary to control a DNS server in order to receive the data; instead, the adversary sends the list of FQDN strings as DNS queries within the network. This attack leverages the common website domain names, which most of them found in whitelist domain names such as ***encoded-data****.cloudfront.net*. Then, within the same network, the attacker captures the DNS requests into the PCAP file and extracts the encoded payload from packet capture into the regular file using *PacketWhisper*. Note that there is not a need to complete successful lookup. The attack's DNS requests look like non-malicious requests since the lengths are short and using a whitelisted domain name. However, it requires sending numerous amounts of DNS requests depending on the file's size. Thus, the traffic analysis module can detect the attack.

Even though the detection system detected all attack scenarios, it identifies 20 cases as a false positive, where the system recognizes them as malicious, but they are not malicious. Interestingly, many big companies leverage the DNS exfiltration to communicate with clients and servers. Thus, based on DNS exfiltration characteristics, the detection system finds these DNS requests as attacks. Table 6.5 shows the most interesting top five false positives. Nintendo sends DNS requests shown in the Table 6.5 every 30 minutes to communicate with Nintendo's servers. Similarly, BitDefender, the Internet security company, uses the DNS exfiltration technique to send file signatures or scan results to update their databases. Other companies such as Google, Sony, Amazon, and others are using this technique for either statistical analysis about the client or send data to their servers.

6.3.2 Results of Experiment 2

The first experimental results show that there are a false-positives in 20 cases. However, most of the cases do have the malicious characteristics of our detection system module, the payload analysis module. Nowadays, companies use the DNS protocol to build communication channels for clients and servers. Thus, we conducted another evaluation of the detection system to resolve this issue. The purpose of the second experiment is to enhance and improve the false-positives rate over time. We solve this by maintaining and updating the whitelists of common domain names found in the network. Thus, this solution requires monitoring the false-positives value of the detection system over time, every week, to whitelist the domain names that are not malicious. The long-term goal is to have few or no false-positives values by keeping update the whitelist of the detection system.

As a result, the evaluation results of the second experiment show that we improved and reduced the false-positives quality, but we may have a new false positive, which means we regularly need to maintain the whitelist domain name. During the second experiment, the detection system identified all attack cases with no false-negative values, but still, there are two new false-positives values.

6.4 Results

This section compares the proposed detection system with *Snort*, the open-source intrusion detection system, to determine which detection system is the better against DNS-based attacks.

6.4.1 Snort

Snort is an open-source intrusion detection and prevention system (IDS/IPS). Thousands of professionals over the world maintain it to provide a secure network for free. *Snort* is a deep packet inspection that uses various rules and signatures to identify and detect malicious network activity, including DNS-based attacks [58].

6.4.2 Snort Setup

We conducted the identical attack scenarios performed against the developed DNS proxy. By default, *Snort* IDS is not configured to monitor DNS protocol and to detect DNS tunneling attacks. We used signatures that publicly available to monitor and identify the DNS tunneling attacks. Then, we compare the results based on confusion matrices values, including the accuracy and missing rate.

We set up *Snort* using SPAN (Switched Port Analyzer) technique, which is also known as port mirroring. SPAN is a monitoring network traffic method that enabled the switch to send a copy of all network packets seen on one port to another port, where the packet can be analyzed. Figure 6.5 shows the exact setup for the experiment. *Snort* is an intrusion detection system that detects malicious network activities based on rules and signatures. These signatures are generated by professionals from what they see in their daily work and experience.

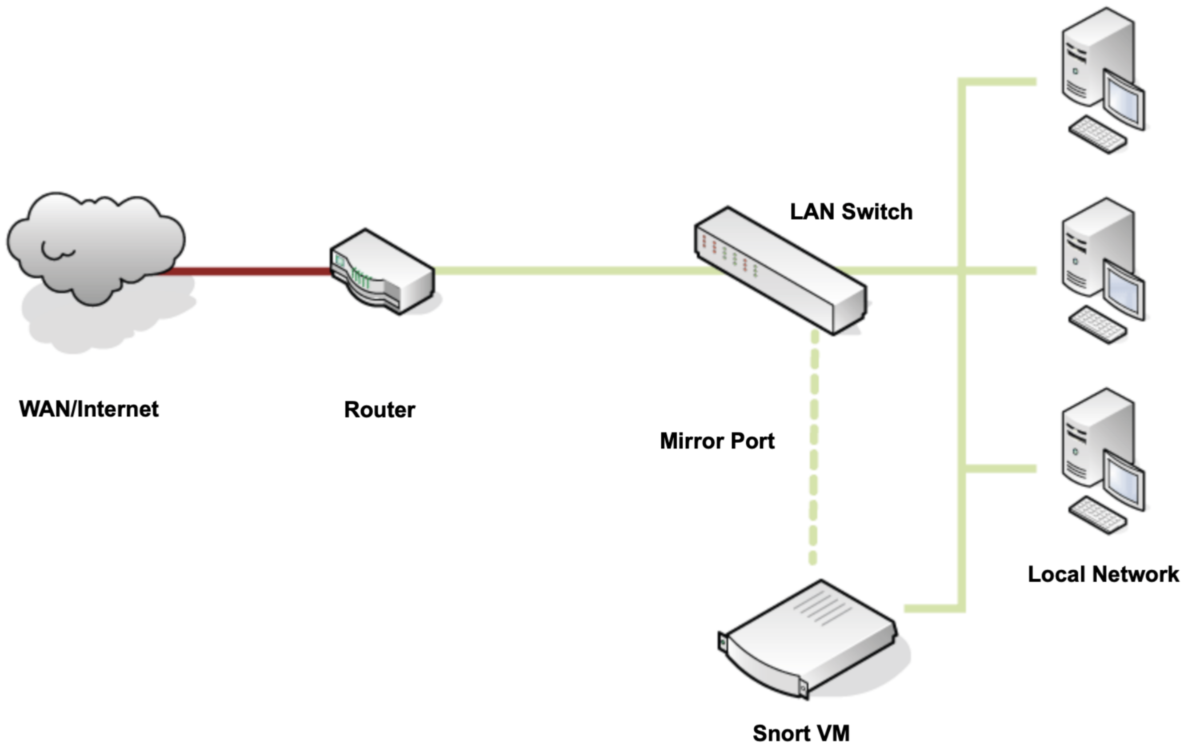
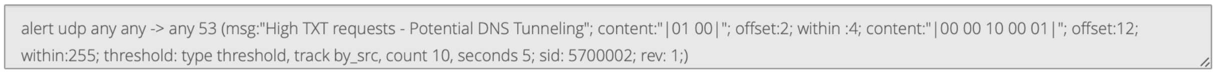


Figure 6.5: Environmental Setup of Snort.

Table 6.6 shows an example of *Snort* signature for TXT DNS type for detecting DNS tunneling attacks. *Snort* monitors a specific UDP packet and header on port 53. *Snort* tracks the sender source IP address by setting a threshold value for TXT DNS type requests and counts 10 DNS requests within 5 seconds. Thus, this can easily bypass the rule if the attacker is aware of it.



```
alert udp any any -> any 53 (msg:"High TXT requests - Potential DNS Tunneling"; content:"|01 00|"; offset:2; within :4; content:"|00 00 10 00 01|"; offset:12;
within:255; threshold: type threshold, track by_src, count 10, seconds 5; sid: 5700002; rev: 1;)
```

Figure 6.6: Snort TXT Signature type for DNS Tunnelling.

Table 6.6: Compared Results between DNS Proxy and Snort

Attack	DNS Proxy	Snort
iodine	Yes	Yes
dns2tcp	Yes	Yes
DET	Yes	No
DNS EX. 1KB	Yes	No
DNS EX. 10KB	Yes	No
DNS EX. 50KB	Yes	Yes
Cobalt Strike	Yes	No
Cobalt Strike-Upload	Yes	Yes
DNSCat2	Yes	No
DNSCat2-Upload	Yes	Yes
PacketWhisper	Yes	No

6.4.3 Snort Results

We conducted different scenarios of DNS-based attacks using *Snort* to detect the same type of attacks within a private network. We configured *Snort* with varying rules that targe *TXT*, *NULL* DNS query types. *Snort* detected only six out of 12 cases of attacks. Thus, *Snort* misses 6 cases of DNS-based attacks that do not validate the rule specifications. *Snort* is capable of detecting DNS-based attacks based on TCP over DNS scenarios and uploading encoded files over the DNS protocol. However, it failed to identify DNS-based attacks based on the DNS exfiltration technique and command-and-control technique, while not having any false-positive cases.

6.4.4 DNS Proxy vs Snort

Table 6.6 shows the attacks detected by the DNS proxy using the techniques described in this work versus *Snort*. There are six attack scenarios that *Snort* is not able to detect. At the same time, the DNS proxy is able to identify and detect all attack scenarios. *Snort* was not able to detect most of the DNS exfiltration attacks with small files as well as the command and control attacks.

Next, Table 6.7 shows three experiments' evaluation results, including true-positives rate, false-positives rate, precision, F1-score, and accuracy. *DNS Proxy - EXP1* is the first experiment as a detection system that does not employ the whitelisted domain names in detecting DNS-based attacks. The *DNS Proxy - EXP2* is the second experiment, the improved version of the detection system that uses the whitelisted domain names to reduce the false-positives rate. *Snort IDS* is the final experiment where *Snort* IDS is in place for monitoring the DNS traffic within the network. We consider the system has higher accuracy with fewer false negatives is a better detection system. The detection system scores 99% accuracy, but it has few false-positive values in the two tries of experiments, DNS Proxy EXPR1 and DNS Proxy EXPR2. Using the whitelisted domain names in DNS EXPR2, we significantly improved the results in the second experiment and decreased the false-positive values from 2% to 0.2%. The *Snort* IDS scores 99% accuracy with no false-positive values. However, *Snort* IDS misses 50% of the attacks results in TPR of 50% and FNR of 50%.

Finally, Table 6.8 summarizes the comparison between the detection system and *Snort* IDS. The table shows the significate difference between the two systems. First, the detection system identified all cases of performed attacks, including the high-throughput and low-throughput techniques. *Snort* IDS was able to detect only 6 cases of the attacks out of 12, mostly the high-throughput. The detection system identified the attacks as soon as it received. While the *Snort* IDS requires 10 to 15 seconds to detect the attack due to the rules' implementation. The detection system works as a DNS proxy that does not require a special configuration or client action. In contrast, *Snort* IDS requires special configuration such as a port mirror. Finally, the detection system provides DNS services, detection, and prevention, while Snort IDS provides detection only.

Table 6.7: Comparison of Three Experiments Results

Experment	TPR	FPR	FNR	PRC	F1	Accurancy
DNS Proxy - EXP1	1.00	0.020	0.00	0.375	0.54	97.9%
DNS Proxy - EXP2	1.00	0.002	0.00	0.85	0.92	99.8%
Snort IDS	0.50	0.00	0.50	1.00	0.66	99.4%

Table 6.8: The Major Difference between the Detection System and Snort IDS

	The Detection System	SnortIDS
Attacks	12/12	6/12
Time	Nearly realtime	10-15 seconds
Configuration	No configuration is needed	Need a Special Configuration
Services	DNS, Protection, Prevention	Detection Only

This section highlights the detection features of the work in this dissertation compared to prior research based on features used to determine the DNS tunneling and DNS exfiltration. After evaluating the proposed detection system performance and validate the method, we compare the proposed method to methods proposed in other published works as shown in Table 6.9. We observed that most of the proposed methods focus on statistical analysis methods, including query length, DNS record types, and volume of DNS requests for detecting the DNS tunneling attacks. On the other hand, a limited number of works have

Table 6.9: Features Used in Detection DNS Tunneling and DNS Exfiltration

Features	[9]	[36]	[38]	[2]	[58]	[28]	Our work
Statistical Analysis		x	x	x		x	x
Query Length			x	x		x	x
Record Types			x	x	x	x	x
Volume of Requests		x			x	x	x
Freq. per domain					x	x	x
Host per domain	x						x
Machine Learning				x			x
Visualization							x
Blacklist server		x		x			x
Time interval			x	x			x
DNS Signatures	x						
High-throughput	x	x	x	x	x	x	x
Low-throughput				x			x
Real-time					x		x
DNS proxy							x

focused on detecting both DNS tunneling and DNS exfiltration using various techniques such as machine learning. Note that none of the previous research works in real-time for detecting against DNS-based attacks. In this work, we combined multiple methods, payload analysis and traffic analysis, to detect both DNS tunneling and DNS exfiltration using visualization and machine learning. We implemented DNS proxy to fully control clients' DNS requests to operate the proposed detection system to work in real-time.

To sum up, we built a detection system that acts as a DNS proxy server to have full control over the DNS requests within the network. At the same time, we conducted various DNS-based attacks using tools to simulate real-world scenarios such as TCP over DNS, DNS exfiltration, command and control channels. The detection system obtains high accuracy with no false-negative rate and fewer false-positive rates. Finally, we compared the detection system results with *Snort* IDS results which also conducted similar previous attacks. *Snort* IDS also scores high accuracy by 99%; however, it could not detect 50% of the attacks. The results indicate that the detection system in this work, the DNS proxy, is better detection against DNS-based attacks.

7 Conclusion and Future Work

This chapter summarizes the work, including the system architecture design, evaluation, and results. It also identifies the limitations of the proposed system. Finally, it describes future work.

7.1 Conclusion

In this dissertation, an original detection system is a network-based solution that detects and prevents DNS-based attacks, including DNS tunneling, DNS exfiltration technique, and command-and-control was developed. The detection system's primary goal is to detect not only high-throughput DNS tunneling techniques but also low-throughput DNS-based attacks. The detection system acts as a DNS proxy server that receives and resolves DNS requests from clients in real-time, as well as analyzing the data offline. The detection system can be broken down into three main components: visualization, machine learning-based detection, and the real-time analyzer modules.

First, we introduced a fast and scalable visualization module to detect and identify DNS tunneling attacks based on the parallel coordinates technique in the visualization module. To the best of our knowledge, this is the first time the DNS tunneling attack has been visualized using parallel coordinates. Network traffic was collected for one week to create a dataset to study the normal behavior of DNS. Six DNS parameters for the parallel coordinates technique were chosen and used to visualize the DNS traffic using the parallel coordinates technique in order to distinguish between normal DNS traffic and malicious traffic containing DNS tunneling attacks. Based on this module, four unique DNS tunneling patterns were identified, each representing various scenarios of DNS tunneling attacks using the parallel coordinates technique. Visualizations were created that are visually very different so that they can be used in the future for detecting DNS tunneling attacks.

Next, the machine learning-based detection module uses real-time passively captured DNS traffic from a network or offline previously captured traffic that is stored in PCAP files to detect DNS tunneling attacks. This module is based on the work is done in Chapter 3, the visualization module. It generates a graphical signature text representation that labels the dataset for machine learning training and testing purposes. The machine learning-based detection module consists of three main components: the Parser, the Analyzer, and the Detector, where the Parser captures and feeds the system with DNS data. The analyzer prepares and analyzes the data for the next step, which is the Detector. The Detector works as a decision-maker built based on supervised learning and pre-trained objects, determining whether the DNS tunneling attack occurs. During the experiment, three months of DNS traffic were captured and collected within a controlled network. In addition, we simulated the DNS tunneling attack by performing the attack within the network. Then we used these captured data and simulated attacks data to train, test, and evaluate our detection module. As a result of this module, the system achieves 99% accuracy to detect all simulated cases of the attacks with no false negatives and a low false-positive rate. Finally, the real-time detection analyzer is a payload analysis module that applies various statistical and analytical functions to the captured data in real-time. Once the DNS request is received, the payload analysis module extracts domain name features to check various malicious characteristics to determine if it is malicious.

The detection system is implemented as a network-based solution of a DNS proxy to have complete control over clients' DNS requests. The detection system targets DNS-based attacks, including TCP over DNS, DNS exfiltration technique, and the command-and-control communication channel. Even if the adversary uses the low-throughput DNS tunneling technique, the detection system can detect and identify the attack using the payload analysis module that uses real-time capture looking for malicious features. The detection system, DNS proxy, combines the three modules discussed before to detect the DNS-based attacks. Furthermore, the DNS proxy consists of two main modes, including the real-time and offline detection modes.

As an overall result, the detection system presented in this work achieves sufficient and satisfying results by obtaining a high accuracy detection rate with no false-negative rate and fewer false-positive rates. During the experiment, we demonstrated that the detection system is capable of providing efficient and scalable services by providing DNS services as well as detection and prevention to clients within the active and busy network.

Finally, we validated the detection system results by comparing the detection system and *Snort* IDS results with the same environment setup and attacks. Even though Snort IDS scores high accuracy, it could not detect all the attack scenarios. It identified only 50% of the performed attacks, of which 6 out 12 total attacks. The results show that the attacker can bypass the *Snort* IDS by performing the low-throughput DNS exfiltration technique. In contrast, our detection system, DNS proxy, attained and scored more reliable and better accuracy, precision, and F1-score performances. F1-score is the harmonic mean of precision and recall models, and it is a method of combining and weighted average of both precision and recall. It provides a better measurement of incorrect cases of detection than the overall accuracy rate. The detection system in this work obtained a F1-score rate of 0.92, while Snort IDS scored 0.66. The detection system, DNS proxy, identified and blocked all 12 cases, including the high-throughput and low-throughput DNS exfiltration techniques. The DNS proxy applies various methods to determine the attacks in real-time and offline detection modes using visualization, machine learning classifiers, and statistical analysis. The results indicate that the detection system in this work, the DNS proxy, is a better network-based solution that monitors and detects the network environment for DNS-based suspicious or unusual activity as well as provides a prevention mechanism.

Based on this work's experimental results, we have designed an original, reliable, and robust detection system that detects DNS-based attacks better than the commonly used open-source *Snort*. Thus, the results indicate that the detection system is applicable and can be employed and integrated into any network infrastructure for individuals or business levels for providing DNS services and protection. System administrators can acquire great benefit from this work by not only DNS services but also protection. The detection system obtained high accuracy and precision rates to identify the DNS-based attacks and no false negatives. The evaluation experiments in this work confirm that the detection system efficiently handled and supplied DNS services to active and busy network DNS requests as well as protected against DNS-based attacks, which helps in blocking the DNS communication channel as soon as it detected.

7.2 Discussion and Limitations

The objective of this work is to detect and block DNS-based attacks. The detection system consists of two main modes: the real-time and offline modes to detects two types of DNS-based attacks, including low-throughput and high-throughput techniques. The reason for designing two methods is to ensure identifying attacks using either one of them or both. In the results chapter, we exhibited a case where the real-time detection missed the attack, *WhisperPacket*. However, the offline mode was still able to detect the attack—having two detection modes makes it more difficult for the adversary to bypass the system's security measures.

We showed that the detection system works efficiently within an active and busy home-lab network. However, it still has some limitations. The DNS proxy, the detection system, is vulnerable to DNS server attacks, including DoS and DDoS, Server hijacking, DNS spoofing, and cache poisoning attacks. Additional work is needed to secure the server. Another limitation is within the caching functionality. The DNS proxy server caches the received DNS query to fast lookup for the next time. The way that we execute the caching is by generating a file that has clients' DNS requests. Therefore, the file size may increase and become large and causing some storage issues that need to be addressed in the future.

7.3 Future Work

The detection system has some limitations that need to be resolved. As we discussed previously, the DNS proxy server is vulnerable to various DNS protocols and server attacks, including DoS and DDoS, Server hijacking, DNS spoofing, and cache poisoning attacks. Solving these issues is planned to be a part of future work to build a comprehensive detection system—for example, the randomization of the source port and transactionID to prevent the DNS cache poisoning attack. Moreover, we need to implement DNSSec, an extra security layer to the existing DNS protocol, to prove the chain of trust using public-key cryptography for authentication and data integrity as well as increase the overall security level of the detection system.

This section highlights future work that can be done to extend this work. First, more machine learning classifiers for the traffic analysis module could be evaluated for both accuracy and speed. In addition, the techniques could be implemented and integrated into home routers or commercial applications starting by designing a module for *Snort*. Finally, the detection system could be deployed into the cloud and provide free cloud-based DNS services and protection to clients.

Bibliography

[1] M. Dooley and T. Rooney, *Introduction to the Domain Name System (DNS).* IEEE, 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8008719

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

[3] R. Elz and R. Bush, "Clarifications to the DNS Specification," Internet Requests for Comments, RFC 2181, July 1997. [Online]. Available: https://www.rfc-editor.org/info/rfc2181

[4] A. M. Kara, H. Binsalleeh, M. Mannan, A. Youssef, and M. Debbabi, "Detection of malicious payload distribution channels in DNS," in *2014 IEEE International Conference on Communications (ICC)*, June 2014, pp. 853–858.

[5] McAfee, "FAQs for Global Threat Intelligence File Reputation," 2019, last accessed 20 August 2019. [Online]. Available: https://kc.mcafee.com/corporate/index?page=content&id=KB53735

[6] P. Satam, H. Alipour, Y. Al-Nashif, and S. Hariri, "DNS-IDS: Securing DNS in the Cloud Era," in *2015 International Conference on Cloud and Autonomic Computing*, Sep. 2015, pp. 296–301.

[7] I. Incapsula, "DNS Flood," 2017, last accessed 06 February 2019. [Online]. Available: https://www.incapsula.com/ddos/attack-glossary/dns-flood.html

[8] Verisign, "Framework For Resilient DNS Security," 2018, last accessed 01 August 2018. [Online]. Available: https://blog.verisign.com/security/framework-resilient-dns-security-dns-availability-drives-business/

[9] G. Farnham Advisor, A. Atlasis, and G. Farnham, "Detecting DNS Tunneling Detecting DNS Tunneling GIAC (GCIA) Gold Certification Detecting DNS Tunneling 2," *sans.org*, 2013.

[10] I. Kim, H. Choi, and H. Lee, "BotXrayer: Exposing Botnets by Visualizing DNS Traffic," *KSII the first International Conference on Internet (ICONI)*, 2009.

[11] H. Choi and H. Lee, "PCAV: Internet Attack Visualization on Parallel Coordinates," in *Information and Communications Security*, S. Qing, W. Mao, J. López, and G. Wang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 454–466.

[12] A. Cuzzocrea and D. Zall, "Parallel Coordinates Technique in Visual Data Mining: Advantages, Disadvantages and Combinations," in *2013 17th International Conference on Information Visualisation*, July 2013, pp. 278–284.

[13] H. Choi, H. Lee, and H. Kim, "Fast detection and visualization of network attacks on parallel coordinates," *Computers and Security*, 2009.

[14] X. D. Hoang and Q. C. Nguyen, "Botnet Detection Based On Machine Learning Techniques Using DNS Query Data," *Future Internet*, vol. 10, no. 5, 2018. [Online]. Available: https://www.mdpi.com/1999-5903/10/5/43

[15] K. R. Romain Fouchereau. (2020, 06) IDC 2020 Global DNS Threat Report . Last accessed 05 August 2020. [Online]. Available: https://www.efficientip.com/resources/idc-dns-threat-report-2020/

[16] M. Conti, N. Dragoni, and V. Lesyk, "A Survey of Man in the Middle Attacks," *IEEE Communications Surveys and Tutorials*, vol. 18, pp. 1–1, 03 2016.

[17] M. McCormick, *Data Theft: A Prototypical Insider Threat.* Boston, MA: Springer US, 2008, pp. 53–68. [Online]. Available: https://doi.org/10.1007/978-0-387-77322-3-4

[18] C. Kim, "Marriott Announces Starwood Guest Reservation Database Security Incident," 11 2018, last accessed 07 August 2019. [Online]. Available: https://news.marriott.com/2018/11/marriott-announces-starwood-guest-reservation-database-security-incident/

[19] V. Goel and N. Perlroth, "Yahoo Says 1 Billion User Accounts Were Hacked," 2016, last accessed 07 August 2019. [Online]. Available: https://www.nytimes.com/2016/12/14/technology/yahoo-hack.html

[20] B. Filkins, "Sensitive Data at Risk: The SANS 2017 Data Protection Survey," *SANS Institute InfoSec Reading Room*, 2017.

[21] W. Lu and A. A. Ghorbani, "Botnets Detection Based on IRC-Community," in *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference*, Nov 2008, pp. 1–5.

[22] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A Survey of Security in Software Defined Networks," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 623–654, Firstquarter 2016.

[23] SonicWall, "SonicWall Cyber Threat Report," 2019, last accessed 05 August 2019. [Online]. Available: https://www.sonicwall.com/lp/2019-cyber-threat-report-lp/

[24] B. Zdrnja, N. Brownlee, and D. Wessels, "Passive Monitoring of DNS Anomalies," in *Proceedings of the 4th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, ser. DIMVA '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 129–139. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-73614-1_8

[25] Y. Jin, K. Kakoi, M. Tomoishi, and N. Yamai, "Efficient detection of suspicious DNS traffic by resolver separation per application program," in *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, Oct 2017, pp. 87–92.

[26] G. Kambourakis, T. Moschos, D. Geneiatakis, and S. Gritzalis, "Detecting DNS Amplification Attacks," in *Critical Information Infrastructures Security*, J. Lopez and B. M. Hämmerli, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 185–196.

[27] A. Merlo, G. Papaleo, S. Veneziano, and M. Aiello, "A Comparative Performance Evaluation of DNS Tunneling Tools," in *Computational Intelligence in Security for Information Systems*, Á. Herrero and E. Corchado, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 84–91.

[28] 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)*, Nov 2018, pp. 1–6.

[29] J. Grunzweig, M. Scott, and B. Lee. (2016, May) New Wekby Attacks Use DNS Requests As Command and Control Mechanism. [Online]. Available: http://bit.ly/1TAYE8j

[30] A. Green. (2017, December) DNSMessenger: 2017's Most Beloved Remote Access Trojan (RAT). [Online]. Available: https://bit.ly/2BxBz6O

[31] K. Alieyan, A. ALmomani, A. Manasrah, and M. M. Kadhum, "A survey of botnet detection based on DNS," *Neural Computing and Applications*, vol. 28, no. 7, pp. 1541–1558, Jul 2017. [Online]. Available: https://doi.org/10.1007/s00521-015-2128-0

[32] L. Krämer, J. Krupp, D. Makita, T. Nishizoe, T. Koide, K. Yoshioka, and C. Rossow, "AmpPot: Monitoring and Defending Against Amplification DDoS Attacks," in *Research in Attacks, Intrusions, and Defenses*, H. Bos, F. Monrose, and G. Blanc, Eds. Cham: Springer International Publishing, 2015, pp. 615–636.

[33] A. Ramachandran, N. Feamster, and D. Dagon, "Revealing Botnet Membership Using DNSBL Counter-Intelligence," in *SRUTI '06: 2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet*, 2006, pp. 49–54.

[34] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a Dynamic Reputation System for DNS," in *Proceedings of the 19th USENIX Conference on Security*, ser. USENIX Security'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 18–18. [Online]. Available: http://dl.acm.org/citation.cfm?id=1929820.1929844

[35] B. Lantz and B. O'Connor, "A Mininet-based Virtual Testbed for Distributed SDN Development," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM '15. New York, NY, USA: ACM, 2015, pp. 365–366. [Online]. Available: http://doi.acm.org/10.1145/2785956.2790030

[36] R. W. Steve Jaworski, "Using Splunk to Detect DNS Tunneling," *SANS Institute InfoSec Reading Room*, 2016.

[37] A. D. Biradar and B. Padmavathi, “BotHook: A Supervised Machine Learning Approach for Botnet Detection Using DNS Query Data,” in *ICCCE 2019*, A. Kumar and S. Mozar, Eds. Singapore: Springer Singapore, 2020, pp. 261–269.

[38] J. Liu, S. Li, Y. Zhang, J. Xiao, P. Chang, and C. Peng, “Detecting DNS Tunnel through Binary-Classification Based on Behavior Features,” in *2017 IEEE Trustcom/BigDataSE/ICESS*, Aug 2017, pp. 339–346.

[39] A. Das, M. Y. Shen, M. Shashanka, and J. Wang, “Detection of exfiltration and tunneling over DNS,” in *Proceedings - 16th IEEE International Conference on Machine Learning and Applications, ICMLA 2017*, 2018.

[40] Y. F. Mohammed and D. R. Thompson, “Visualization of DNS tunneling attacks using parallel coordinates technique,” in *Security, Privacy, and Anonymity in Computation, Communication, and Storage - 12th International Conference, SpaCCS 2019, Atlanta, GA, USA, July 14-17, 2019, Proceedings*, ser. Lecture Notes in Computer Science, vol. 1161. Cham, Switzerland: Springer, 2019, pp. 89–101. [Online]. Available: https://doi.org/10.1007/978-3-030-24907-6_8

[41] K. Born and D. Gustafson, “NgViz: Detecting DNS Tunnels through N-Gram Visualization and Quantitative Analysis,” *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research - CSIIRW ’10*, 2010.

[42] D. E. Eastlake and C. W. Kaufman, “Domain Name System Security Extensions,” *RFC*, vol. 2065, pp. 1–41, 1997.

[43] G. Ateniese and S. Mangard, “A new approach to DNS security (DNSSEC),” in *ACM Conference on Computer and Communications Security*, 2001.

[44] J. Bau and J. C. Mitchell, “A Security Evaluation of DNSSEC with NSEC3,” *IACR Cryptology ePrint Archive*, vol. 2010, p. 115, 2010.

[45] A. Revelli and N. Leidecker, “Introducing Heyoka: DNS Tunneling 2.0,” 2009, last accessed 07 February 2019. [Online]. Available: http://heyoka.sourceforge.net/Heyoka-SOURCEBoston2009.pdf

[46] Panda - Python Data Analysis Library. Last accessed 06 November 2018. [Online]. Available: https://pandas.pydata.org/

[47] The Go Programming Language. Last accessed 06 February 2019. [Online]. Available: https://golang.org/

[48] The R Project for Statistical Computing. Last accessed 06 November 2018. [Online]. Available: https://www.r-project.org/

[49] H. Target, “Download Top 1 Million Sites,” 2020, last accessed 06 August 2020. [Online]. Available: https://hackertarget.com/top-million-site-list-download/

[50] K. Born, “A passive approach to network-wide covert communications,” *Black Hat*, 2010.

[51] D. G. Kumar Ahuja, “Evaluation Metrics for Intrusion Detection Systems-A Study,” *International Journal of Computer Science and Mobile Applications*, vol. 11, 06 2015.

[52] E. Skoudis. (2012, 02) The six most dangerous new attack techniques and what’s coming next? Last accessed 02 September 2018. [Online]. Available: https://blogs.sans.org/pentesting/files/2012/03/RSA-2012-EXP-108-Skoudis-Ullrich.pdf

[53] J. Guy. (2009, 02) DNS Part II: Visualization. Last accessed 02 January 2021. [Online]. Available: http://armatum.com/blog/2009/dns-part-ii/

[54] P. Mockapetris. (1987, 11) DOMAIN NAMES - CONCEPTS AND FACILITIES . Last accessed 20 February 2021. [Online]. Available: https://tools.ietf.org/html/rfc1034

[55] V. Horenbeeck. (2006, 06) DNS tunneling . Last accessed 20 February 2019. [Online]. Available: http://www.daemon.be/maarten/dnstunnel.html

[56] J. Gervais. (2018, 08) PacketWhisper Exfiltration Toolset . Last accessed 12 December 2020. [Online]. Available: https://github.com/TryCatchHCF/PacketWhisper

[57] D. Schaper. (2015, 03) Pi-hole website . Last accessed 20 February 2021. [Online]. Available: https://pi-hole.net

[58] M. Roesch, “Snort - Network Intrusion Detection & Prevention system,” 1998, last accessed 06 February 2021. [Online]. Available: https://www.snort.org