DOMAIN NAME SYSTEM (DNS) SECURITY: HEALTH MEASUREMENT AND INTRUSION DETECTION

Thesis

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

SANJAY

Reg. No.: 177039MA500

DEPARTMENT OF MATHEMATICAL AND COMPUTATIONAL SCIENCES

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

SURATHKAL, MANGALORE - 575025

MAY, 2022

DECLARATION

By the Ph.D. Research Scholar

I hereby *declare* that the Research Thesis entitled **DOMAIN NAME SYSTEM (DNS) SECURITY: HEALTH MEASUREMENT AND INTRUSION DETECTION** which is being submitted to the **National Institute of Technology Karnataka, Surathkal** in partial fulfillment of the requirements for the award of the Degree of **Doctor of Philosophy** in **Mathematical and Computational Sciences** is a *bonafide report of the research work carried out by me.* The material contained in this Research Thesis has not been submitted to any University or Institution for the award of any degree.

Sanjay

Sanjay

Reg. No.: 177039MA500

Department of Mathematical and Computational Sciences

Place: NITK, Surathkal.

Date: 17 MAY 2022

CERTIFICATE

This is to *certify* that the Research Thesis entitled **DOMAIN NAME SYSTEM (DNS) SECURITY: HEALTH MEASUREMENT AND INTRUSION DETECTION** submitted by **Sanjay**, (Reg. No.: 177039MA500) as the record of the research work carried out by him, is *accepted as the Research Thesis submission* in partial fulfillment of the requirements for the award of degree of **Doctor of Philosophy**.

[illegible]

(Dr. Pushparaj Shetty D)

Research Supervisor NITK Mangaluru

R. Balaji

(Dr. Balaji Rajendran)

Research Co-Supervisor C-DAC Bengaluru

[illegible]

Chairman - DRPC

Chairman
DUGC / DPGC / DRPC
Dept. of Mathematical and Computational Sciences
National Institute of Technology Karnataka, Surathkal
MANGALORE - 575 025

DEDICATION AND ACKNOWLEDGMENT

I want to convey my heartfelt thanks and appreciation to the many people I have collaborated with over the last several years for their unwavering support and encouragement, which has helped me reach this point in my doctorate studies.

Dr. Pushparaj Shetty D., my supervisor (Department of Mathematical and Computational Sciences, National Institute of Technology Karnataka), thank you for your patience, advice, and support. Your words of encouragement and insightful, thorough comments have meant a great deal to me. I am glad that you accepted me as a student and continued to believe in me throughout the years.

Dr. Rajendran Balaji, my co-supervisor, deserves credit for his constant mentoring and never-ending supply of intriguing assignments. His unpretentious attitude to science and study is inspiring, and his extensive expertise and thorough editing have been beneficial to me.

I thank the Research Progress Assessment Committee (**RPAC**) members for their continuous support and encouragement. I convey many thanks to all fellow doctoral students, teaching faculties, and non-teaching staffs in the Department of Mathematical and Computational Sciences for encouraging to pursue hardwork and their cooperation.

Finally, I should mention my source of inspiration, my wife, **Mrs. Sandeepa Barasa**, for her unwavering mental support during the difficult journey. Many thanks to her for making my life more beautiful and green.

ABSTRACT

DNS is an essential service for the smooth functioning of all Internet services including web applications, email services, messaging services, online social networks, etc... that works by resolving an alphabetic hostname to an IP address. Every network communication usually begins with a DNS mapping of the given domain name to the IP address specifically accessible by the program. Many programs stop working as soon as this mapping service is inaccessible. As a result, adversaries are keen to deny this service via bug exploits, exploiting the vulnerabilities, or by circumventing the protocol standard function. In such instances, the attackers may use DNS server settings that lack sufficient security hardening and to do things like moving DNS zones, altering DNS resolvers to report fake IP addresses to divert customers, divert web and email traffic, or execute deadly DNS amplification attacks.

DNS also plays a significant role in the overall user experience of Internet services. However, it is mostly forgotten, and is often discovered without adequate protection, or running older software versions, or is entirely insecure. Since most organizations are unaware that DNS is a key attack vector, DNS-based attacks and exploitations occur. The system of DNS could appeal to an attacker for disruptive operations such as network footprinting, downloading malicious software, contact with command and control servers, data exfiltration out of a network, and DNS-based reflective amplification DDoS attempts. In some cases, DNS applications face many security issues such as DoS/DDoS attacks on DNS servers, DNS cache poisoning, NXDomain (Non-existent domain), MITM (Man-in-the-Middle) attacks, and DNS ID spoofing. As a result, monitoring DNS traffic for threat security is very important.

This research aims to provide two novel methods for securing DNS infrastructure, i.e., DNS health measurement and DNS intrusion detection. As part of the research goal, we compare the DNS query resolution latency in IPv4 versus IPv6 network stacks by setting up a three-level DNS hierarchy for the forward lookup tree and a four-level DNS hierarchy for the reverse lookup tree on a dual IP stack that includes ROOT on top-level, TLD on second, SLD on third, and subdomains on the fourth level. We also set up a dual-stack-based recursive resolver.

In this thesis, we offer a unique empirical technique for measuring the health of authoritative DNS servers — a key, essential, and significant component of the DNS infrastructure. DNS software weaknesses, DNS latency comparison with ICMP latency, and DNSSEC validation are three new parameter classes that are proposed and evaluated for effective assessment of the health of authoritative DNS servers. The proposed methodology can be extensible across the components of the entire DNS infrastructure and could be used to analyze, identify, and prevent DNS abuse regularly.

We present - DNS Intrusion Detection (DID), a system integrated into SNORT - a prominent open-source IDS, to detect major DNS-related attacks. We have developed novel IDS signatures for various tools used in the DNS tunneling, DNS amplification, and DNS DoS attacks. We identified the above DNS attacks carried out by different tools available on the Internet using our method. DID observed a high detection rate and a low false-positive rate during testing.

KEYWORDS: DNS; DNS Health; DNS IDS; DNS Tunneling; DNS Amplification; DoS Attack; SNORT; IPv6 DNS Query Latency; DNS Hierarchy Testbed.

Contents

List of Figures

List of Tables

ABBREVIATIONS

DNS	Domain Name System
DNSSEC	Domain Name System Security Extensions
IPSEC	IP Security
TCP/IP	Transmission Control Protocol / Internet Protocol
TLD	Top Level Domain
TTLD	Third Top Level Domain
SLD	Second Level Domain
NS	Name Server
DID	DNS Intrusion Detection
RR	Recursive Resolver
IETF	Internet Engineering Task Force
BIND	Berkeley Internet Name Domain
ICANN	Internet Corporation for Assigned Names and Numbers
DNSEXT	DNS Extensions working Group
MeNSa	Measuring Name System Health
EDNS0	Extension Mechanisms for DNS
RTT	Round Trip Time
ICMP	Internet Control Message Protocol
SOA	Start of Authority
NXDOMAIN	Non-Existent Domain
RFC	Request for Comments
IPSEC	Internet Protocol Security
AXFR	DNS Zone Transfer Query
TSIG	Transaction Signature
PRNG	Pseudorandom Number Generator
QID	Query Identification
MITM	Man in the Middle Attack
VP	Vantage Point
DDoS	Distributed Denial of Service

Chapter 1

Introduction

The Internet has grown drastically from past two decades, almost everything in this world is connected in one or another way to the Internet which includes public and private organizations in the sectors of government, mobile communication system, health sector, nuclear and power energy, emergency services, agriculture, commercial institutions & banking system, aviation transportation system, judiciary & law makers, and police & defense. There are almost 5 billion users connected on the Internet that is 64 % of the world's population. Around 1.8 billion websites are hosted over the Internet, and currently, over 30 billion devices are connected on this massive network [1] [2].
The functionality of today's Internet is completely dependent on Domain Name System - DNS, as it is one among the most significant components of the Internet and one of the key reasons the Internet is as vast as it is today. DNS is essential for the Internet to function by mapping memorable website names or domain names to network layer addresses. If the DNS system fails on the Internet, no websites will be accessible using their names, the email system will stop functioning, and all the applications/services that use domain names become inaccessible. Thus DNS is treated as the most critical component of overall Internet infrastructure, and DNS has been abused to perform large-scale attacks. In the recent past, various attacks have been carried out on the DNS. In the absence of DNS, all the essential services needed in everyday life tasks, e.g., commerce, financial, transportation, and healthcare, become unavailable for the end-users. The DNS becomes the most targeted Internet protocol by the malicious users for causing damage and gaining personal benefits; therefore, DNS faces many security abuses such as DNS tunneling for data exfiltration, command & control (C2) attacks,

reflected amplification for generation DDoS traffic, DNS cache poisoning, DNS spoofing & flooding, NXDOMAIN & NSNXDOMAIN attacks, DNS hijacking, Phantom Domain attack and Botnet-based Distributed Denial of Service (DDoS) attacks [3].
DNS could be appealing to an adversary for malicious activities such as network reconnaissance, malware downloads, contact with command and control servers, or data transfers out of a network. As a result, monitoring DNS traffic for threat security is very important. Since DNS is the most critical component of the Internet and intranet, the security of DNS infrastructure is essential. This research work aims to focus on the challenges in DNS security and provides two methods for securing DNS infrastructure, i.e., measuring the health of DNS servers and DNS Intrusion Detection (DID) System; it also illustrates the process of setting up IPv6 aware DNS hierarchy testbed and its comparison evaluation for IPv4 and IPv6 protocols.

1.1 Research Motivation

DNS was designed by Paul Mockapetris in the early 1980s [4] with little consideration for security. Today, it becomes a pivotal service at every network core, making it a popular target for malevolent attacks. Because DNS capabilities has largely stayed constant, there are a number of inherent vulnerabilities exists, and flaws in the DNS protocol's architecture are frequently abused in data leakage attacks.
As DNS has become the most critical infrastructure of the Internet, its failure impacts the whole functioning of the Internet; websites cannot be accessed with the URL, email flow will be interrupted, and all IoT devices will malfunction to connect to the Internet. Therefore, evaluating the health condition of DNS servers has become a pressing issue that must be addressed now. There are two general types of attacks on DNS, i.e., those that intend to disable the DNS service and those intended to impact the data. One well-known example for the former is a Denial of Service (DoS) attack on the DNS servers. The second major type of attack is much more clever, and it seeks to change the information of the DNS response to send the user into an alternate, often compromised location. DNS spoofing and cache poisoning is the example of such attacks. Shockingly for around 36 years of the historical backdrop of the Internet, DNS was aimlessly

conniving in a wide range of malicious activities.

On 21 October 2002, all 13 root nodes of DNS observed massive DDoS traffic for an hour. The one-hour assault was not observable to the normal end client, was done by sending huge ICMP echo requests to the root DNS nodes using hundreds of controlled bots. This Botnet based DDoS attack did not impact the Internet but merely slowed the sections of the web; in this attack, one of the root name servers observerd around 80 Mbps of network traffic, ten times more than usual [5].

In 2007 three out of 13 root DNS servers were disturbed by the botnet attack, which caused little disruption in the Internet service, and some corporate clients have even seen that the attacks were reoccurring. The first wave of attack lasted two and a half hours, while the second wave lasted five hours. The G and L root servers suffered badly due to lack of anycast support and experiencing network failure that dropped some website access attempts. 20 Gbps network traffic was observed during the attack on each server [6] [7] [8].

On 19 March 2013, the Spamhaus website was put down by DNS reflected amplification attack. IP addresses of the Spamhaus website were spoofed for sending DNS queries to over 30000 open DNS resolvers asking all records of ripe.net domain name. The public recursive resolvers responded with a DNS zone file, collectively generating approximately 75 Gbps of attack traffic at the Spamhaus website that chocked the network access at the website, resulting inaccessibility of the website [9].

The two consecutive waves of botnet-based DDoS were observed on many root DNS nodes, which lasted two and half hours and one hour on 30 November and 1 December 2015. The root DNS observed high DNS query requests, around 5 million per second for two domains, www.336901.com and www.916yy.com [10].

DYN DNS service provider in Europe was the main target of IoT botnet-based DDoS attack which happened multiples times in a day on 21 October 2016. Mirai malware was used to infect around 0.1 million IoT devices and converted them to the Bot to enslaved in the Mirai Botnet, which collectively flooded port 53 of DYN's DNS server with TCP SYN request that generated around 1.2 Tbps network traffic, preventing end-users from accessing more than 1,200 domains of DYN DNS server and the attack prevented traffic

from reaching DYN's customers websites including Twitter, Spotify, Netflix and Amazon [11] [12].
The traditional cyber security solutions such as IDS and IPS are limited in scope to defend against DNS-based attacks because they do not address a wide range of security threats related to the DNS ecosystem (as these solutions do not protect from DNS amplification and tunneling attacks generated from a wide range of tools). There are much research conducted for studying and analyzing the DNS ecosystem; correspondingly, the mainstream of the resolutions proposed (DNSSEC, TSIG, etc.) hitherto is designed primarily to target cache poisoning and MITM attacks, but very few studies have been concerned with measuring the health of DNS, and DNS intrusion detection. These studies and their research gaps are described in Chapter 2 and these studies has the common research gaps (for example, no research work done in the area of DNS health measurement that accounts for all of the health indicators listed by ICANN and added by the MeNSa framework and in addition to this no research work is done in the area of DNS intrusion detection which accounts all the tools used for DNS amplification and tunneling attacks to detect these attacks). Furthermore, no research work is done on the commission of at least three levels of DNS hierarchy with dual-stack for both forward and reverse lookup zone and its latency evaluation related to DNS queries. Therefore it shows the importance of DNS security for Internet infrastructure which motivates researchers in their research domain.

1.2 Problem Definition

Over the past two decades, as the Internet has enormously evolved, the DNS infrastructure has been abused many times, leading to some portions of the Internet is not accessible or feeling delayed. Any assault on the DNS system has the potential to do significant harm to the millions of users who rely on it to function in the background discreetly. There are many added security protocols like TSIG, DNSSEC, etc., evolved to secure DNS infrastructure. Furthermore, many security appliances like DNS firewall and UTM has been developed to intercede DNS. Nevertheless, these appliances do not protect from all possible DNS attacks; and there has not been much study done on intrusion detection systems for the sole purpose of protecting DNS infrastructure.

Against this background, in this thesis, we aim to find a methodology to secure DNS infrastructure by:

1. Finding the methodology to measure the health of the DNS server.
2. Creating IDS signature of DNS amplification, DNS tunneling, and DNS DoS attacks and propose a security solution called "DNS Intrusion Detection (DID)."
3. As a part of research statements 1 and 2, there is a need to establish an IPv6-aware dual-stack enabled DNS hierarchy testbed setup and conduct a DNS query latency evaluation.

1.3 Research Objective

The main objective of this thesis work is to contribute to the field of knowledge by developing DNS security solutions i.e. DNS health measurement and DNS intrusion detection. The designed solutions are meant to protect DNS infrastructure from various known threats related to DNS. This thesis work also aims to provide a novel methodology to conduct the first-ever study of DNS query latency comparison for IPv4 and IPv6 protocols on a live testbed that emulates the three-level of DNS hierarchy for the forward lookup and four-level DNS hierarchy for the reverse lookup tree.
In order to find the answer to the research topic, few questions are formulated and each of these questions is part of the research topic. The research topic formulated is: "Domain Name System (DNS) Security: Health Measurement and Intrusion Detection." The research question that formulated based on identified research gaps are:

Question 1. What types of attacks target DNS infrastructure and use DNS infrastructure to attack other systems, and how?
To secure DNS, we need to list various threats and classify those threats into attacks on DNS infrastructure and attacks exploiting the DNS infrastructure.

Question 2. What is the methodology to find the health of the DNS server?
Find out the parameters that identify the health of the DNS server and health measurement method based on these parameters.

Question 3. What are the attack signatures for DNS amplification, DNS tunneling, and DoS attacks?
We need to determine the traffic characteristics of these attacks, create an appropriate IDS signature to detect these attacks and use these signatures in the existing IDS i.e. SNORT for detecting intrusion for DNS.

Question 4. How to set up IPv6 aware dual-stack enabled DNS hierarchy testbed?
To answer the above three questions, we need to create three levels of IPv6-aware dual-stack DNS hierarchy testbed that can be further used for DNS health measurement and DID. We also evaluate the DNS query latency for two network layer protocols i.e, IPv4 and IPv6.

1.4 Thesis Contributions

The DNS is the key component of the Internet and needs to be protected from various well-known threats. Every communication that includes a domain name begins with a name resolution to determine the IP address associated with it. Almost all applications and protocols that are engaged in network communication use the DNS significantly. As a result, someone can inevitably notice the flaws in DNS and exploit them at some time. This Ph.D. thesis aims to provide the following contributions in the field of DNS security:

1. We propose a novel list of parameters specifically for determining the health of authoritative name servers. The aim is to understand the general behavior of authoritative name servers, detect sluggishness in their performance, and arrive at a score of their health through the identified list of parameters. The effectiveness of identified parameters is evaluated by devising the corresponding probing algorithms and experimenting with them among the authoritative name servers serving the world's top 500 domains. The proposed approach can be used periodically to assess and take necessary measures to protect authoritative domain name servers from abuse.

2. We built unique IDS signatures for the following DNS-based attacks: DoS, Amplification, and Tunneling, and added them to the original rule-set file of SNORT IDS to detect DNS-based intrusions. The proposed method effectively detects empirical DNS assaults carried out using a variety of well-known tools available on the Internet. DID was shown to have a high detection rate and a very low false-positive rate during testing.

3. We aim to conduct the first-ever study of DNS query latency comparison for IPv4 and IPv6 protocols on a testbed that emulates the DNS hierarchy. We also highlighted the implementation of DNS hierarchy (root DNS, TLD, SLD, etc.) using a simplified method on dual-stack. This research work also provides a baseline for the configuration of root DNS, TLDs, authoritative domains, and recursive resolver over IPv4 and IPv6 network-layer protocols.

1.5 Thesis Structure

The remaining of this thesis is as follows:
Chapter 2 introduces some key concepts and technologies related to DNS and illustrates various components of the DNS ecosystem. The DNS namespace, hierarchical structure, DNS server classifications, query types, DNS cache, lookup procedure, DNS message format, record types, and zone file structure are all illustrated in this chapter. We also discuss DNS abuses, divided into two categories: attacks against DNS infrastructure and attacks that exploit DNS infrastructure. Additionally, we provide a comprehensive view of previous work aiming to analyze DNS health measurement, DNS intrusion detection, and IPv6 aware DNS hierarchy test-bed setup and its evaluation. It also includes a gap analysis that shows what information is accessible, what information is lacking, and how to fill those gaps in this and future studies.
Chapter 3 describes our novel methodology for monitoring of DNS health, as well as the health metrics discovered throughout the research. We also explain the several algorithms used to evaluate each identified parameters. The experiment was conducted on the name servers of the top 500 domains in the world and we highligted the cirtical problem areas that needs to be addressed by DNS system.

Chapter 4 focuses solely on the necessity for an intrusion detection system in the DNS environment. It also discusses the various tools to carry out DNS tunneling and DNS amplification attacks and describes the signature established in this research for detecting attacks from these listed tools. We also include the results of an experiment conducted to assess the effectivness of DID with modified IDS signatures by compairing with default SNORT settings.

Chapter 5 illustrates the process of setting up a DNS hierarchy on a dual-stack for both forward and reverse lookup trees. The approach for evaluating DNS latency for the DNS hierarchy is also discussed. The experiment is conducted from several locations across the world, and the results are described.

Finally, we conclude our thesis in Chapter 6 by summarizing the research results and proposing the future scope of work.

Chapter 2

Background and Literature Survey

In computer networking, the network equipment does not interact between them by name as people do. Computers and other comparable devices interact and identify themselves via a network using numbers such as IP addresses. People, on the other hand, are used to referring to things by their names rather than their numbers, whether they are speaking to another person or identifying a country, location, or object. As a result, to overcome the communication gap between computers and people for enabling communication easier, networking scientists evolved the Domain Name System (DNS), which translates human-friendly names such as www.nitk.ac.in to network numbers such as 218.248.46.85. Therefore when a user types a website address or domain name into a web browser, DNS converts the name to a number. If a person desired to visit a specific website, they put website name on the browser, such as www.cdac.in. In order to get a web page, If users already knew the IP address, the user no longer has to write www.cdac.in; instead, the user just needs to type in the website's IP address. We may input the domain name, and DNS will convert it to an IP address, because we are not used to memorising and dealing with numbers, especially when there are billions of websites on the Internet. When a user types www.cdac.in into a web browser, a query to a DNS server is sent, asking for the IP address of the www.cdac.in. The DNS server then searches a database on the Internet for a matching IP address for www.cdac.in, and if it finds one, it resolves www.cdac.in to 196.1.113.45. The web browser then requests the web page from the web server running on 196.1.113.45, and it opens the web page after getting the response back from the web server. This complete process is illustrated in Figure 2.1.

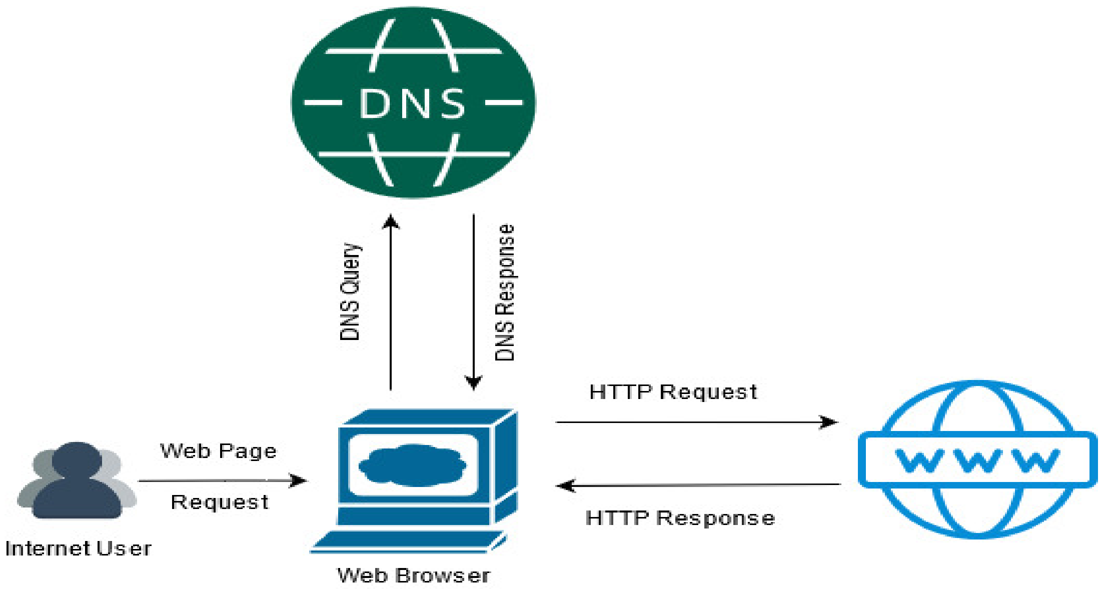


Figure 2.1 Working of DNS

2.1 DNS History

In the 1970s, mapping human-readable hostnames to numerical addresses was developed with Advance Research Project Agency Network - ARPANET, the forerunner of today’s Internet. On the ARPANET network, a single file called "HOSTS.TXT" was used on a centralized computer operated by NIC Stanford Research Institute (SRI) for mapping of IP address to the domain name [13]. Users would call SRI staff during working hours to add an entry to the hosts file, and they would manually add the host and its corresponding numeric address to the file. For name resolution, this file must be distributed to all systems in the network. The central system performed well for more than a decade. The file size had grown much larger than anticipated in 1982, and the pace of change increased as the network grew and the centralized system’s limit became apparent. It was recognized that a centralized, manually modified host file would not be scalable. Paul Mockapetris was given the task of developing an automatic naming scheme by John Postel of the University of Southern California. Mockapetris was meant to find a middle ground between five different technological solutions, but instead, he came up with a new one, i.e., DNS. In 1987 the DNS specifications were updated and formalized in RFC 1034 and 1035, resulting in the basic protocol still in use today [14] [15].

2.2 DNS Fundamentals and Concepts

The core principles of the Domain Name System are discussed in depth in this section. We use the DNS terminology defined by the IETF DNS operation working group in this section and the rest of this chapter.

2.2.1 DNS Namespace

The DNS namespace is a tree structure that manages public hostnames on the Internet. Each node has a textual label as well as one or more DNS resource records (RR) that define the domain. The label, as well as the labels of its parent nodes, are separated by a dot (as in "cdac.in") in the domain name. The core concept in the DNS is the domain name; a standardized ASCII character string is used to describe a domain name. A domain name comprises one or more labels separated by dots, and the length of a label is limited to a maximum of 63 octats [76]. The top-level domain (TLD) is at the far right, and the labels below it, from right to left, are lower in the namespace hierarchy. Each label is referred to as a subdomain of the label immediately above it. There are 127 hierarchical levels in DNS; the letters A-Z, a-z, the digits 0-9, and the hyphen (-) can appear on labels. The case of the labels does not matter, i.e., www.cdac.in and WWW.CDAC.IN are equal.

As shown in Figure 2.2, in a textual hostname, the presence of the root domain is sometimes indicated by a single dot at the end of the name, but this dot is often omitted. The root domain, i.e., dot (.), terminates the domain name and always is addressed as an empty label and represented by HEX value 0x00. When parsers (web browsers, dig commands etc.) of DNS messages come across the root domain, they must stop processing the domain name. The fully qualified domain name, which is often abbreviated to FQDN, refers to the full domain name, which includes all labels that make up the name, including the root label, TLD label, SLD label. This term (FQDN) is often interchanged with the shorter term "domain name." In this thesis, the word "domain name" refers to a fully qualified domain name (FQDN).

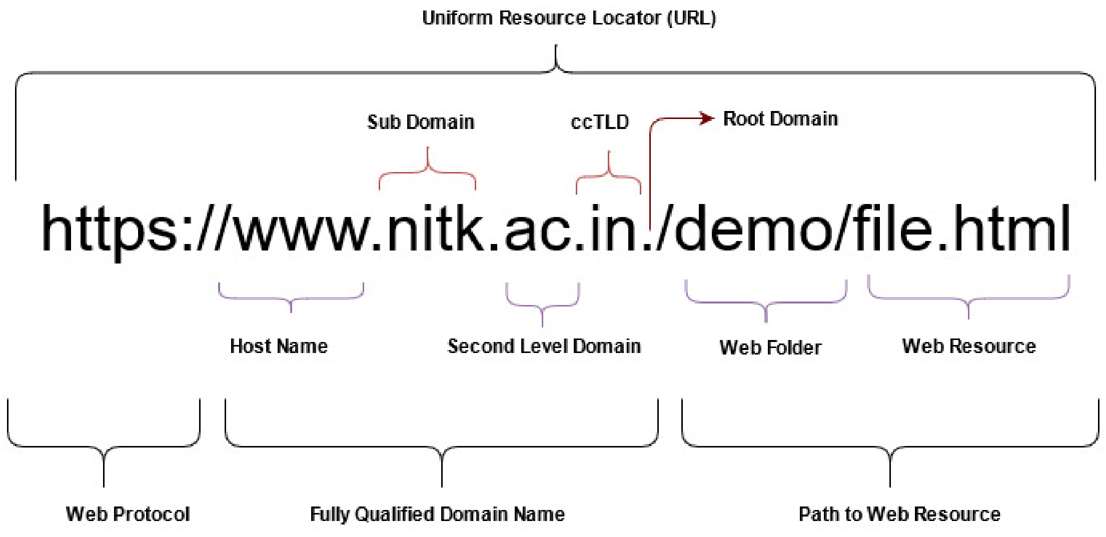


Figure 2.2 The Domain Namespace Sample

2.2.2 DNS Hierarchy

DNS follows a hierarchical database system distributed across the globe with millions of servers nodes over the Internet, and many of these nodes are authoritative for specified domain names. However, to convert a given domain name to an IP address, it can obtain information from other servers in the DNS infrastructure. It implies that while a recursive DNS may not have all of the records needed to resolve a domain name like "www.cdac.in," it can figure out who to ask. Through port 53, DNS Protocol employs both TCP and UDP as transport layer protocols. TCP is typically used for excahnge of domain information between primary and secondary DNS servers, whereas UDP is used for DNS server-client requests and answers.

As shown in Figure 2.3, the root domain is the highest level of the DNS hierarchy. It is responsible for delegating administrative responsibility for TLDs, the next section of a domain name. Over 1498 top-level domains have been assigned to the root zone database in the DNS root zone [16].

TLDs are in the next level of the DNS hierarchy, followed by the root domain and positioned on the extreme right of the domain name. Second-level domains - SLDs are the next level down in the tree after TLDs. Subdomains or subsequent level domains and hostnames are found under SLDs. Internet's forerunners, now known as the Address and Routing Parameter Area-ARPA, formed a special TLD i.e ARPA, responsible for the reverse lookup process, i.e., resolving IP address to the domain name.

There are mainly three types of TLDs, i.e., gTLD, ccTLD, and infrastructure TLD.

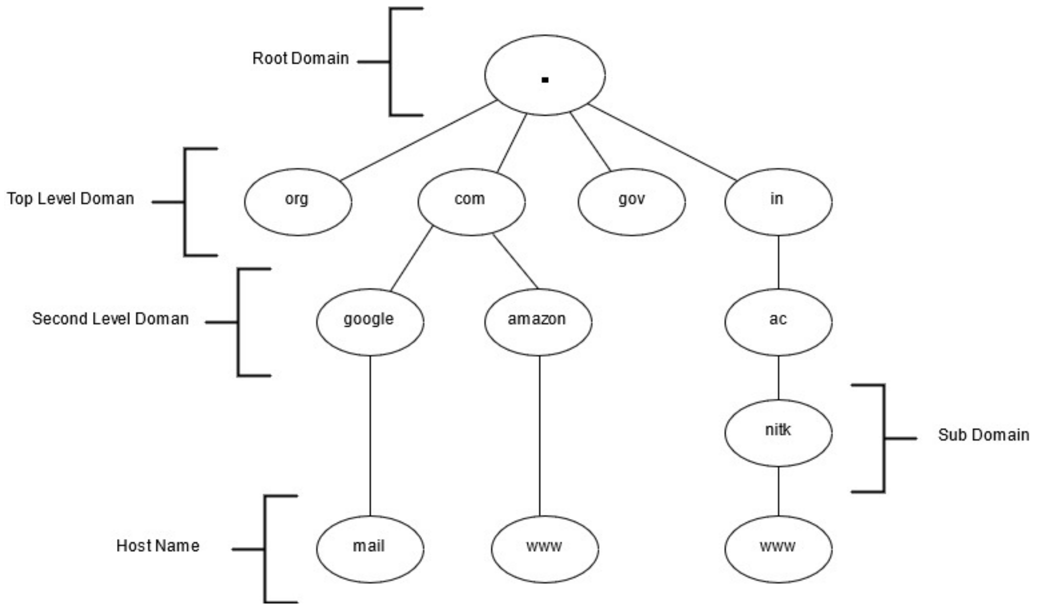


Figure 2.3 A Sample DNS Hierarchy

1. gTLD - A gTLD (generic top-level domain) is a domain class that has three or more characters. The word "generic" is in the name of gTLDs because they have generic organization descriptors. These domain extensions are also not geographically restricted, and anyone from anywhere in the world can register them. Seven generic top-level domains were established before the establishment of ICANN in 1998 [17] during the early stages of the Internet's growth, as shown in Table 2.1. The number of generic top-level domains (gTLDs) is constantly increasing. The gTLDs that have recently been introduced are referred to as "new gTLDs."

2. ccTLD - ccTLDs (country code top-level domains) are usually reserved for countries, territories, and sovereign states, identified with a two letters country code. The examples of ccTLDs are “in” (India), “uk” (United Kingdom"), “us” (United

States of America), “cn” (China), “tk” (Tokelau) etc. The criteria for ccTLDs are set by the domain name regulation corporation in each country. There are currently 248 ccTLDs in the root zone, and the IANA root database currently contains 1498 TLDs as of Jan 2022.

3. Infrastructure Top-Level Domain (arpa) - This category has only one TLD, which is arpa. This TLD is mostly used for technical network infrastructure administration. With two subdomains, in-addr.arpa and ip6.arpa, this TLD is used for reverse search of IPv4 and IPv6 addresses. Although the term was initially an abbreviation for the Advanced Research Projects Agency, a US funding agency that created the ARPANET core network, it was later renamed the Address and Routing Parameter Area.

Table 2.1 Generic Top Level Domain Name

com	This generic TLD, which stands for “commercial,” was originally intended only for commercial use, but limitations on its use have since been removed. Com domains may be used for a numerous purposes, including personal biographies.
net	The “net” extension is derived from the term “network.” This generic top-level domain was created to be used by networking companies and Internet Service Providers (ISPs). However, the limits on its use have since been removed, allowing businesses from all sectors to use this gTLD.
org	The term “organization” has been shortened to “org.”. Org was created to be used by charities and non-profit organizations. There are currently no limits on who can register this gTLD.
int	The “int” is acronyms to “international”, this gTLD is only available to organizations, offices, and services recognized by a treaty signed by two or more countries.
mil	Abbreviated to “military” specific to the US military, this gTLD is restricted to United States Department of Defense branches, programs, and agencies.
gov	The term “government” is shortened to “gov”, and this gTLD is only available to US government departments and agencies, as well as eligible federal, city, and local government agencies.
edu	The gTLD domain “edu” came from “education”, is a sponsored top-level domain. The domain was created to provide a domain name hierarchy for organizations that focus on education, even though they are not affiliated with the US.

2.2.3 DNS Server Classifications

The DNS comprises many name servers located worldwide and is arranged in a hierarchical order to achieve its functionality. There is no single server on the Internet that holds DNS data for all domain names to IP address translations. Instead, records for a particular domain are stored on a specific DNS server. It can also forward DNS questions for tranlations that the server does not have, to another DNS server that does. As a result, there are three types of DNS servers.

Table 2.2 List of Root DNS

Root Server Name	Letter	Operator	IP Address(IPv4 and IPv6)	Instances
a.root-server.net	"A"	Verisign	198.41.0.4 2001:503:ba3e::2:30	16
b.root-server.net	"B"	USC-ISI	199.9.14.201 2001:500:200::b	6
c.root-server.net	"C"	Cogent Communications	192.33.4.12, 2001:500:2::c	12
d.root-server.net	"D"	University of Maryland	199.7.91.13 2001:500:2d::d	156
e.root-server.net	"E"	NASA Ames Research Center	192.203.230.10 2001:500:a8::e	254
f.root-server.net	"F"	Internet Systems Consortium, Inc.	192.5.5.241 2001:500:2f::f	247
g.root-server.net	"G"	Defense Information Systems Agency	192.112.36.4 2001:500:12::d0d	6
h.root-server.net	"H"	U.S. Army Research Lab	198.97.190.53 2001:500:1::53	8
i.root-server.net	"I"	Netnod	192.36.148.17 2001:7fe::53	69
j.root-server.net	"J"	Verisign, Inc.	192.58.128.30 2001:503:c27::2:30	118
k.root-server.net	"K"	RIPE NCC	193.0.14.129 2001:7fd::1	75
l.root-server.net	"L"	ICANN	199.7.83.42 2001:500:9f::42	187
m.root-server.net	"M"	WIDE Project	202.12.27.33 2001:dc3::35	6

2.2.3.1 Root DNS Server

A root DNS server is at the top of the DNS hierarchy and is the authoritative name server for the root domain indicated by a dot (.). 12 distinct companies own and manage the 13 root DNS servers with the letters A through M shown in Table 2.2. However, it should be noted that all A to M servers are identical, with the same root zone file containing the list of authoritative name servers for each TLD. There are many root server instances (copies of a root server) worldwide to increase query performance. The ICANN-operated L root server, for example, has 187 instances around the world. As a result, the root name servers are essential components in the DNS query resolution phase, as the resolution begins from them. Volunteer initiatives from a wide range of organizations currently provide root name server operations.

2.2.3.2 Authoritative DNS Server

DNS services are typically provided by Web hosting companies or DNS hosting companies on the Internet. They are usually in charge of the authoritative DNS servers for a domain name, which clients ask. Since they are the authoritative source for the domain in question, authoritative DNS servers serve the actual response: the final translation of the domain name to the IP address. Master and slave authoritative name servers are the different kinds of authoritative name servers. The master authoritative name server maintains the original DNS records and database. In contrast, the slave authoritative name server maintains a duplicate copy, with the slave contacting the master for database updates.

2.2.3.3 Recursive DNS Server

A recursive DNS server accepts user requests for website names or domain names and checks authoritative DNS servers records for the corresponding IP address. Since recursive server caches the result obtained from an authoritative server, recursive DNS servers are expected to support authoritative DNS servers, which would otherwise be unable to handle the massive amount of traffic generated by user requests. The first servers contacted as a result of a request are recursive DNS servers. The recursive server looks for the mapping in its cached response from an authoritative server with a valid TTL, and it sends the response back to the client with mappings. If the recursive server does not have the IP address, it looks through the authoritative DNS server hierarchy. ISPs usually operate recursive DNS servers for their clients, but certain organizations have their recursive resolvers. Corporations such as Cloudflare (1.1.1.1), Google (8.8.8.8), CDAC (223.31.121.171), and other organizations operate several free public recursive DNS servers.

2.2.4 DNS Query Types

A client sends a request for the resource records to the DNS server in the form of DNS query which is usually sent for getting IP address with respect to a domain name. The two ways for resolving a host or domain name to an IP address using the domain name scheme are recursive and non-recursive (iterative) queries.

2.2.4.1 Iterative DNS query

In an iterative query, the recursive DNS server queries all the levels in the DNS hierarchy until it receives an answers for the given query from each level. Therefore it always does iterations of queries starting from root DNS on level 0 and followed by TLD on level 1, SLD on level 2 and so on, as shown in Figure 2.4.
For iterative queries, the DNS server responds with a referral to another DNS server in the hierarchy that may respond to the given query. The root DNS server and TLD DNS server always react to iterative queries, and they are set up to only allow non-recursive or iterative querying.

2.2.4.2 Recursive DNS query

The client requires a complete response to a recursive query, so a recursive DNS server repeatedly queries until it receives the complete response on behalf of the client. In this case, the client expects an IP address for the specified domain name without additional work. During this process, every DNS server holds the query, and it will ask to next DNS server in the DNS hierarchy level, as shown in Figure 2.5.

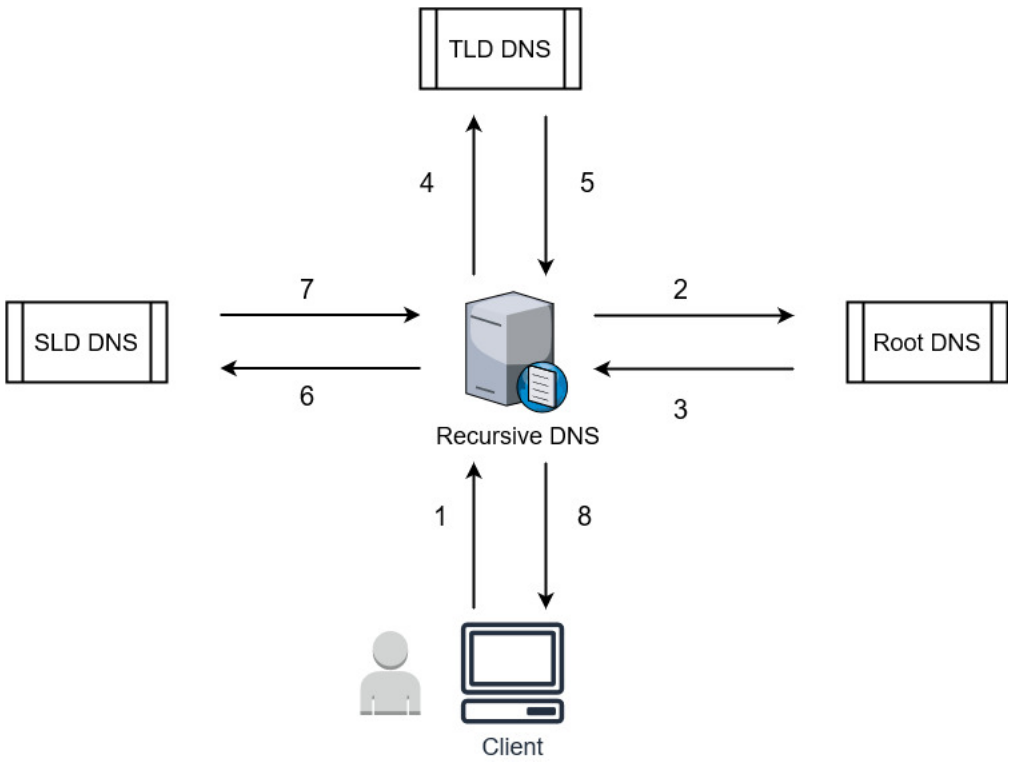


Figure 2.4 An Example of Typical Iterative Query

2.2.5 DNS Cache

The goal of caching is to store data for a short period of time in order to increase the efficiency and reliability of data queries. DNS caching is the practise of storing data closer to the asking client in order to resolve DNS queries quicker and prevent extra searches farther down the DNS lookup chain, resulting in faster load times and lower bandwidth/CPU use. DNS data can be cached in a number of locations, each of which will keep DNS records for a set amount of time based on a time-to-live value (TTL). DNS records are cached by default in modern web browsers for a fixed period. When a request for a DNS record is made, the browser cache is the first place where the requested record is looked for. The status of the DNS cache on the firefox web browser can be checked by typing "about:networking/#dns" into the address bar and "chrome://net-internals/#dns." for the chrome web browser. The purpose is obvious: the closer DNS caching is to the web browser, the lesser computing steps are required to search the cache and make the proper requests to an IP address. If the browser does not get lookup from its DNS cache, it will send a DNS query to an operating system level DNS resolver

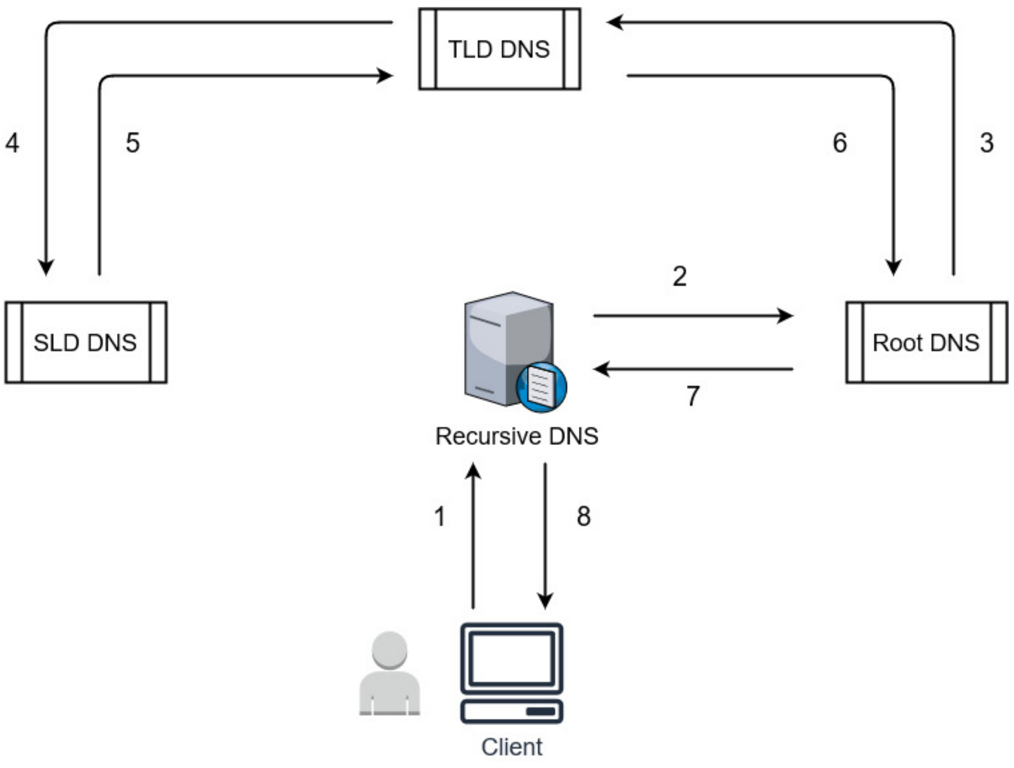


Figure 2.5 The Recursive Query

typically known as "stub resolver" or DNS client. When an application sends a DNS query to "stub resolver", it first looks in its cache to see if the record exists. If the record is not in its cache, it forwards the query to the local recursive server or ISP recursive server. The recursive server then checks its DNS cache to resolve the requested DNS query, and if found, the response is returned to the stub resolver.

2.2.6 DNS Lookup Process

When an Internet user opens a web browser and types a website name like www.example.com into it, the web browser contacts the stub resolver i.e. DNS client which looks up the network address for the specified website name on a recursive resolver assigned by Internet Service Provider (ISP), the request is then forwarded to the recursive resolver of the ISP, if the recursive resolver cannot find a response in its DNS record and cache data, the request is then forwarded to DNS servers in the hierarchy until it obtains the network address of the given website name or NXDOMAIN response if the domain does not exists. The produced answer will be sent back to the query's originator. For finding the IP address of "www.cdac.in," the recursive DNS server begins by choosing one among the 13 root name servers. The root server returns an address that points to the ".in" TLD name server. The recursive DNS server then queries the ".in" TLD name server, which response with an address for the authoritative server for the "cdac.in" domain, which is then contacted by the recursive DNS server, which returns the IP address for the specified FQDN as shown in Figure 2.6.

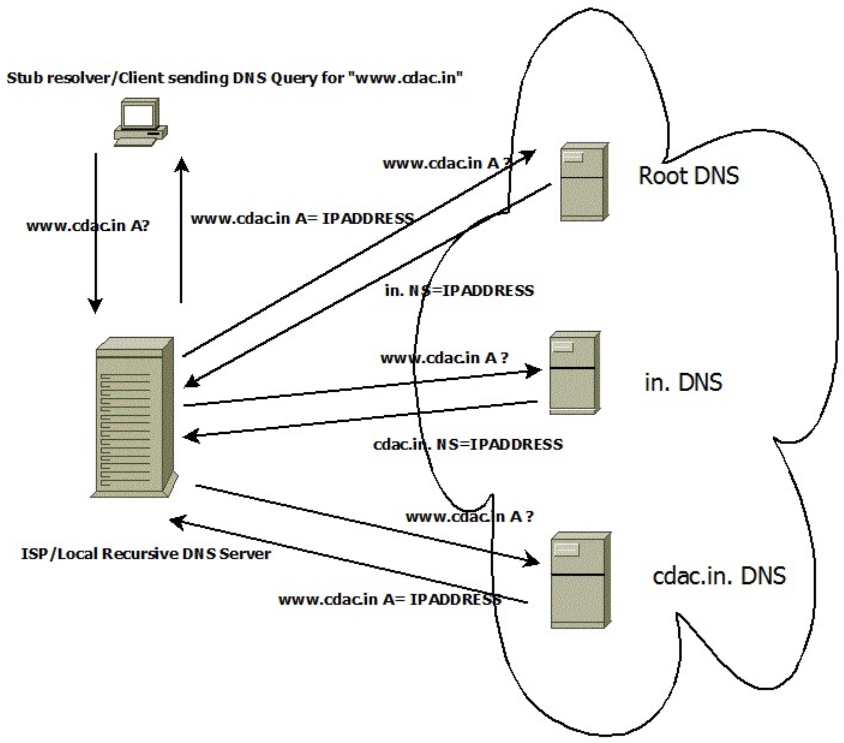


Figure 2.6 DNS Lookup Procedure for "www.cdac.in"

2.3 DNS Protocol

DNS queries are served using the UDP transport layer protocol on port 53. UDP is chosen for DNS requests and answers because it is quick and has a minimal overhead. A DNS query consists of a single UDP request from the DNS client and a single UDP response from the DNS server. When a DNS response is greater than 512 bytes or a DNS server is performing activities like zone transfers, i.e. sending DNS records from the main to the secondary DNS server, TCP is used instead of UDP to provide data

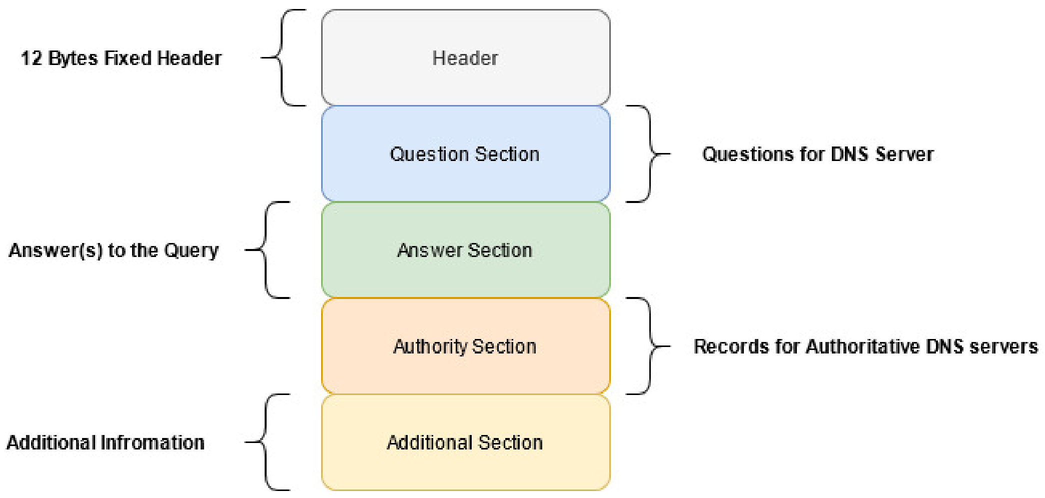


Figure 2.7 DNS Message Format

integrity checks.

2.3.1 DNS Message Format

In DNS, the information exchange between client and server is facilitated using DNS request and reply transmission. Each request and reply follow the same common pattern, with up to five separate sections containing data as shown in the Figure 2.7. The Header section and the Question section are normally included in both queries and answers [18]. Every DNS query and response message has 12 bytes DNS header, which contains a unique ID representing each query/response, questions and answers count, and additional information. The header section is followed by the Question and Answer section. For DNS query, Answer section contains blank and number of answer section in DNS header becomes 0. In the case of DNS reply, the Answer section includes the IP address and domain name mapping. Each request and reply follow the common structure, except the length of the reply packet is always bigger than the request or query.

2.3.2 DNS Header Format

Since essential control fields are held in the header of any protocol, it is the most important part of any message. The header portion of DNS messages contains many important control flags and information about which of the other parts of the DNS message are being utilize in the communication. Analyzing the header helps to grasp a few of the complexities of how DNS messaging works. Figure 2.8 depicts the structure of the header segment used in every DNS packets. These fields are used in different ways by the client and server communications. Each header field is described as follows:

Query ID - It is called query identifier for the client; the client uses this 16-bit field

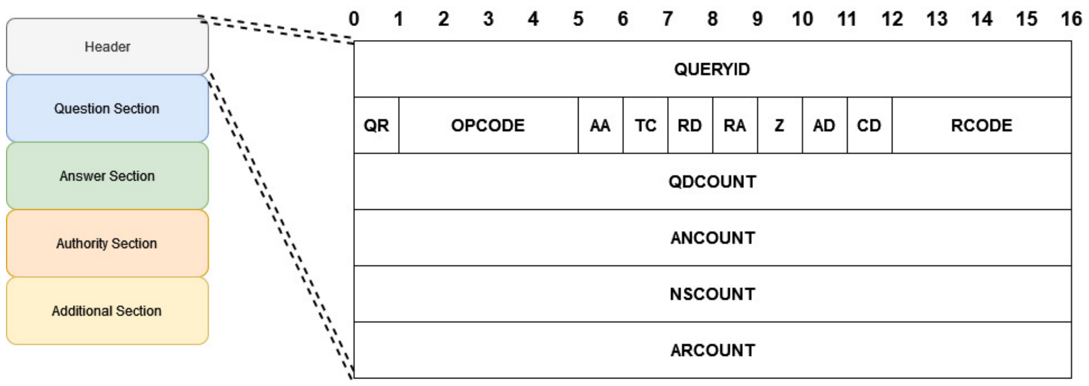


Figure 2.8 DNS Header Format

to fit the answer to the question. Every time the client sends a query, it uses a different identification number. In the corresponding answer, the server repeats this number.

Query/Response Flags - It is a 16-bit field which includes the subfields as described as follows:

1. QR - This is a one-bit field that makes a distinction between DNS queries and their responses. When the query is produced, it is set to 0; when the query is modified to an answer by a replying server, it is set to 1.

2. OPCODE - This is a 4-bit subfield that defines the type of query the carried by the message. This parameter is defined by the creator of the query and replicated into the response unaltered. Table 2.3 shows various OPCODE used while crafting a DNS query.

3. AA - This one bit is called an authoritative answer flag and is used when the server responds to the client. This bit indicates that the DNS server's response is authoritative (bit is set to 1) for the zone in which the domain name is mentioned

Table 2.3 List of OPCODE for DNS Query

Query Message	Query Code	Description
QUERY	0	It is used for a standard DNS query for asking the IP address of a domain name.
INVERSE	1	An inverse DNS query for asking domain name of an IP address
STATUS	2	A Server status query
(Not Used)	3	Currently not used, reserved for future use
NOTIFY	4	A primary authoritative server uses a special message form introduced by RFC 1996 to notify a secondary authoritative server that data for a zone has changed, prompting the secondary server to request a zone switch.
UPDATE	5	RFC 2136 introduced a spatial message form to enforce ’dynamic update,’ enabling resource records to be selectively added, removed, or modified.

in the Question section. If the response is non-authoritative, then this bit is set to 0.

4. TC - The truncation flag is a single bit that, when set to 1, indicates that the message was truncated because its size exceeded the limit permissible for the transport mode utilised. This bit often indicates that the message was transmitted using UDP and was too big to fit, since TCP has no message length restriction whereas UDP messages are restricted to 512 bytes.

5. RD - This bit is called a Recursion desired flag and is used by the client. When this bit is set in a DNS query, it demands that the server receive the query attempt to address it recursively if it supports the recursive resolution. In the response, the value of this bit remains unchanged.

6. RA - This is a single bit referred to as recursion available flag used in the server response. In an answer, set to 1 or cleared to 0 to show if the server that produced the response supports recursive queries. The system that sent the query will then save this information for future use.

7. Z - This one bit is always set to zero and is retained for future usage.

8. AD - The DNSSEC protocol uses this one bit for indicating whether the response contains authenticated data.

9. CD - This one bit used by DNSSEC protocol for DNSSEC checking needs to be disabled if set.

10. RCODE - This is a response code flag of 4 bits and always set to zero in query, then updated by the responding server to express the query's results. This field is used to show whether the question was successfully answered or whether an error occurred. Table- 2.4 shows possible response codes when the server replies.

QDCount - It is 2 bytes field denoted by question count, which indicates the number of questions in the "Question Section" of the DNS query message.
ANCount - The number of resource records in the message's Response section is defined here in 2 bytes.
NSCount - The number of resource records in the message's Authority section is defined here in 2 bytes.
ARCount - In the Additional portion of the message, this field specifies the number of resource records.

2.3.3 DNS Questions Format

This section consists of one or more question records and is always present on both query and response messages. In most DNS queries, at least one element in the Question part explains what the client in the exchange is trying to figure out. If required, these values are transferred in their original form to the answer message for the client's reference. Figure 2.9 depicts the format for each entry in a DNS message's Question section. Following are the fields of question section:
QNAME - This field is called a Question Name. The object, domain, or zone name that is the subject of the query is denoted with regular DNS name format in this field. A domain name is made up of a sequence of labels that start with an octet length and end with the same number of octets. For the null label of the root, the domain name ends

Table 2.4 List of RCODE for DNS Response

Response Message	Response Code	Description
NOERROR	0	No error occurred
FORMERR	1	An issue with the query's construction prevented the server from responding to it. The query format is unknown to the server, so the server cannot reply to the response.
SERVFAIL	2	The server was unable to answer to the query related to a server-side issue.
NXDOMAIN	3	The domain does not include the name mentioned in the query.
NOTIMP	4	The server does not support the type of query that was sent.
REFUSED	5	The query was denied by the server.
YXDOMAIN	6	Name that should not exist does exist
YRRSET	7	RRset that should not exist does exist
XRRSET	8	The RRset does not exist
NOTAUTH	9	The server receiving the query is not authoritative for the zone specified.
NOTZONE	10	Used for a name mentioned in the query is not within the zone specified in the query.

with a zero-length octet. This field is variable in length.

QTYPE - A two-byte code that specifies the type of the query. Typical query types are A, AAAA, MX, NS, PTR, CNAME, TXT etc.

QCLASS - A two-byte code that Specifies the resource record's class, usually 1 for Internet ("IN") class.

2.3.4 DNS Answer Format

This section contains the response of the DNS server to the client or stub resolver. There are one or more resource records in this section, and it only appears in response from the DNS server. Figure 2.10 shows DNS answer format, and the details of each field are as follows:

NAME - The domain name for which the query was sent to the DNS server and in the same format as the QNAME in the question section.

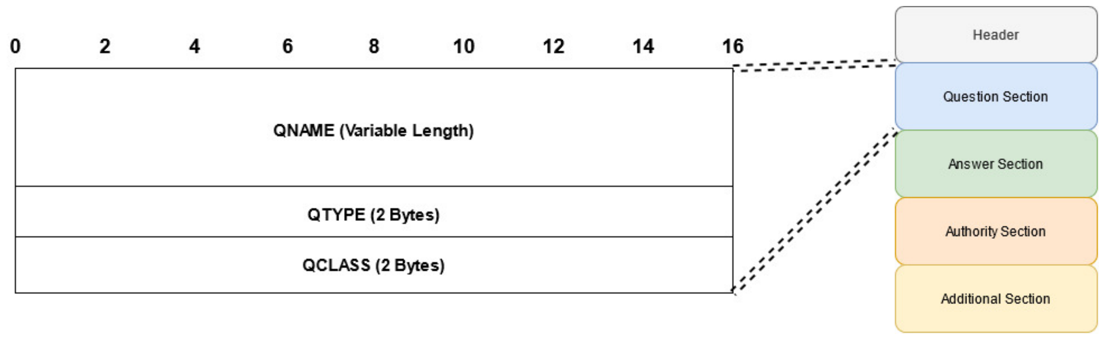


Figure 2.9 DNS Question Format

TYPE - The kind of data (A,MX,NS etc) in the record is specified by two bytes.
CLASS - The record's Class is provided by two bytes, and the value is always "IN."
TTL - These four bytes determine how long (typically in seconds) the resource record can be cached before being discarded by a recursive resolver or anyone who caches these records. Zero values indicate that the current transaction can only use the resource record and should not be cached.
RDLENGTH - Two bytes specify the length of the RDATA field.
RDATA - It is variable length bytes that describe the data in the response of the query. The format of this data varies based on the TYPE of resource record and CLASS. RDATA field is a four-octet ARPA Internet address if the TYPE is A and the CLASS is IN.

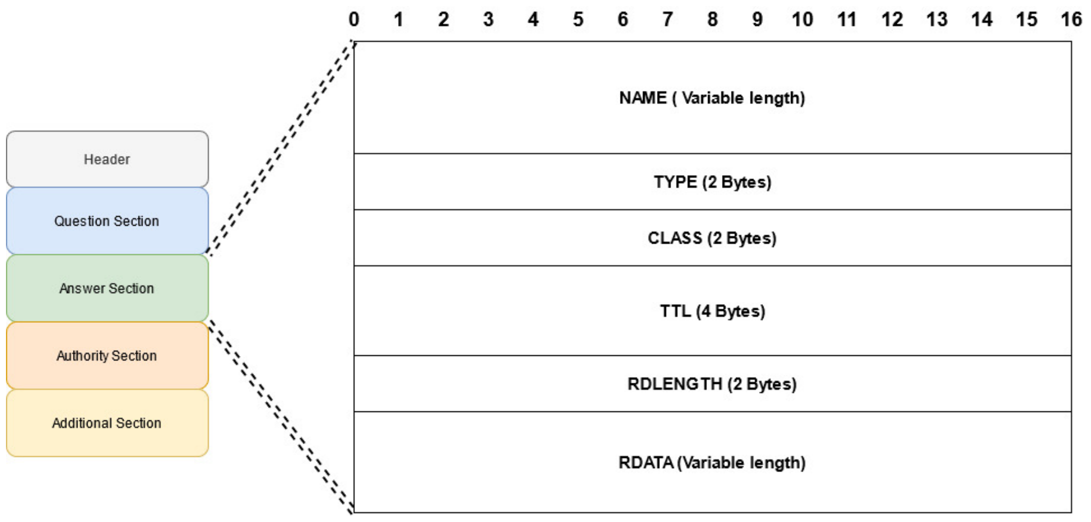


Figure 2.10 DNS Answer Format

2.3.5 Authoritative Section

There are one or more resource records in this section. It only appears in the response message from the DNS server. This section provides information about one or more authoritative servers for the domain name in question. This section contains the same fields as the Answer section.

2.3.6 Additional Information Section

Like Answer and Authoritative section, There are one or more resource records in this section, and it only appears in the response message from the DNS server. This section provides additional information that may help the resolver. This section also contains the same fields as the Answer section.

2.4 DNS Record Types

DNS records, often referred to as zone files, are directives that reside in authoritative servers which include information about a domain, such as its IP address and how to manage requests for that domain. DNS servers create a DNS record to include essential information about a domain or hostname, including its current IP address. These records are made up of a sequence of text files written in DNS syntax. DNS syntax is simply a series of characters that function as commands to the DNS server. TTL, or time-to-live, is a property of all DNS data records that shows how soon a DNS server can update that information. The Table 2.5 lists the most popular DNS record types. For an Internet user to reach a website using a domain name, all domains must have at least a few basic DNS records, and some extra records serve additional purposes.

2.5 DNS Record Syntax

The resource records are coded in ASCII and organized according to a simple scheme. Each DNS record is listed in a distinct line in the zone file. The Figure 2.11 shows a standard structure for the records.

A space separates each field, and even some fields may become optional. Additional fields occur in many types of records (e.g., MX). The details of each field are as follows:

Table 2.5 Most used DNS Resource Records

Record Name	Type	Description
A	IPv4 Address	They are used for translating a fully qualified domain name (FQDN) to the IPv4 address that corresponds to it.
AAAA	IPv6 Address	They are used for translating a FQDN to a corresponding IPv6 address.
SOA	Start of Authority	This record contains authoritative information about a DNS zone, such as the primary DNS server, the domain administrator's email address, the domain serial number, and multiple timers for zone refresh. It also contains the information needed between primary and secondary DNS servers for zone transfer.
MX	Mail Exchange	It defines the mail servers that handle incoming emails for a certain domain and correctly routes emails between domains.
NS	Name Server	It specifies the authoritative name servers for a particular domain name. A domain usually has multiple NS records, which unfold primary and backup nameservers for that domain.
PTR	Pointer to domain	It converts the IP address of a network to a FQDN. PTR records are used in reverse DNS lookups, where DNS queries begin with an IP address and end with a domain name lookup.
CNAME	Canonical Name	It is DNS aliases where a record maps to another name for cosmic or functional reasons. It is mostly used for name-based virtual hosting and for separating different services offered by a single host.
TXT	Text Record	Provides a way for DNS to store arbitrary strings for text or other data. It is also used to fight email spam and for automatic service discovery on private networks.
ANY	All Records	All resource records of all types know to the authoritative server are returned.
OPT	Optional	This is a pseudo-record type that is required for EDNS protocol to work along with DNS.

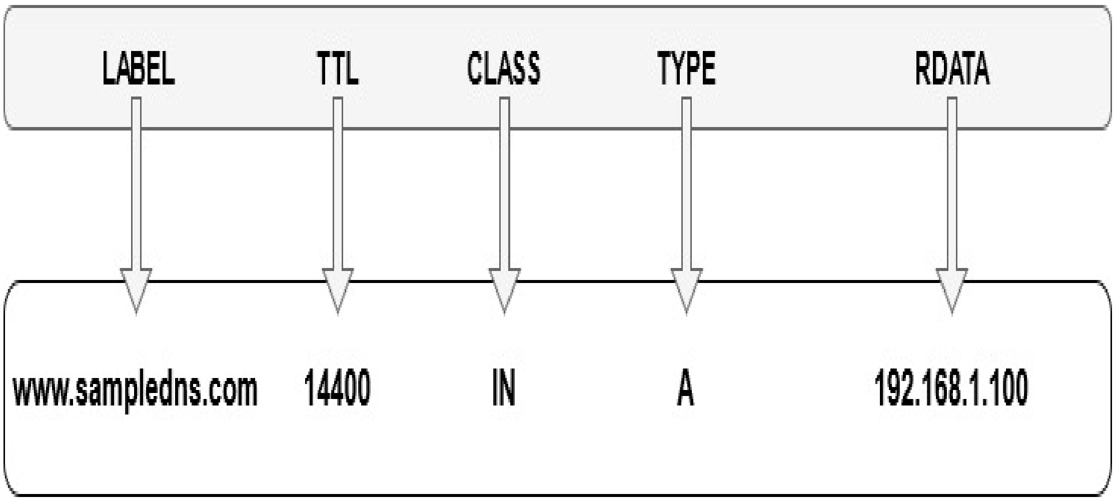


Figure 2.11 Resource Record Format Example

LABEL - It is a FQDN that is entered in the browser by the users.
TTL - An optional information and stands for "time to live," which refers to the amount of time (in seconds) that a record can be kept in the cache.
CLASS - This is an optional field that defines a class of the resource record. Many classes exist for DNS records, but the most commonly used class is the Internet (IN).
TYPE - A zone file contains a variety of resource records, as shown in Table- 2.5.
RDATA - The details that allow the domain name to be resolved.

2.6 DNS Zone File

All DNS records are specified in the zone file, which is nothing but a simple text file. Clear criteria must be followed in order for the data to be interpreted properly by the DNS application. Otherwise, the DNS would be unable to run, resulting in the SERV-FAIL error message being shown to the recipient. As a result, it is important to follow a certain structure. The zone name is defined first, followed by the TTL in certain situations. The majority of DNS documents are stored in zone files. A zone in DNS refers to a specific organizational unit and is assigned to each DNS server. Suppose a client wants to know a mapping of a certain domain. In that case, it looks for the necessary records in the zone files and redirects the request to a lower-level server in the DNS hierarchy before it reaches the final destination. A single line is used per record, and the record ends with a line break. However, if a record needs to span several lines, brackets

are used to the file. Semicolons can be used whenever comments are to be made.
The resource record syntax discussed earlier can be used in the zone file for mentioning zone information in the form of resource records, as shown in the following sample zone file:

A Sample Zone File

```
$Origin sampledns.com
$TTL 4400
@               IN SOA sampledns.com. root.sampledns.com. (
                                07052021 ; serial
                                2D ; refresh
                                3H ; retry
                                2W ; expire
                                1H ) ; minimum
             IN       NS        nameserver.sampledns.com.
             IN       A        192.168.2.111
             IN       MX        10       mailserver.sampledns.com.
nameserver             A         192.168.2.111
mailserver             A         192.168.2.100
www                    A         192.168.2.101
```

2.7 DNS Threats Classification

Since the DNS system is an essential element of the Internet infrastructure while still having several security flaws, it becomes a popular target for hackers. This is a serious cyber security problem that leads to the malfunction of the Internet. Unfortunately, it happened several times in the past three decades. DNS is vulnerable to a variety of attacks if it is not installed and configured properly. A healthy and reliable DNS is required for the Internet to run properly. DNS uses simple, easy to use and connection-

less transport layer protocol, i.e., UDP. There is no algorithm in UDP for checking that the sending packet's source is the source that it seems to be. As a result, an attacker will eavesdrop on UDP/IP packets and create a fake packet that appears to be sent from another source, called IP address spoofing. The packet's recipient has no assurance that the originating IP address in the receiving packet is the true source of the packet [19]. Since UDP applies no overhead to IP packets, DNS queries on UDP have much lower latency than queries on TCP. DNS uses UDP rather than TCP because the mapping between host name and IP address is critical to latency. Remote attackers can see a lack of routine DNS audits as an enticing opportunity to launch malicious attacks against the victim networks. As a result, it is important to keep monitoring DNS servers and traffic to counter DNS attacks.
DNS attacks are categorized as "Attacks against DNS Infrastructure" that include attacks against DNS services like an attack against authoritative and recursive servers. The other category is "DNS attacks exploiting DNS infrastructure," which includes reflection, amplification, domain hijacking, redirection, DNS malware, data exfiltration, tunneling, etc.

2.7.1 Attack Against DNS Infrastructure

These attacks are aimed at the DNS infrastructure itself to either make the DNS service inaccessible or corrupt the reply given by the DNS servers. DNS is made up of two distinct components: authoritative servers and recursive servers, and that each component is vulnerable to different types of attacks. Figure 2.12 depicts different threats that could occur during the DNS query resolution workflow. The description of each attack is as follows:

2.7.1.1 Man in The Middle

A man in the middle attack is a popular cyber security attack that helps an adversary to listen a conversation between two systems. A MITM attack vector is very popular among LANs, and it works on almost all network communications that are not encrypted or authenticated. Here the adversary must place themselves on a network between the two network systems, and then the adversary can intercept communica-

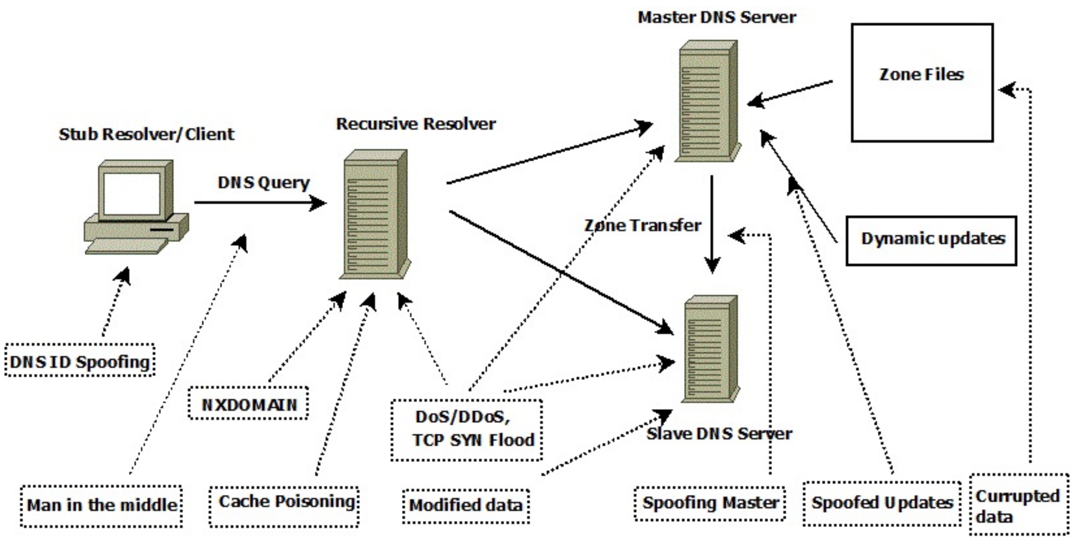


Figure 2.12 Attacks Against DNS Infrastructure

tions traveling in any direction and modify or drop messages. In the case of DNS, the adversary will typically place themselves between the stub resolver and the recursive resolver, then change the DNS response to have a new IP address for the requested domain name, essentially rerouting the user anywhere they wish [20]. MITM is usually done using ARP cache poisoning in a LAN environment; as a result, an adversary will capture and modify the communication between the stub resolver and recursive resolver, making a message seem legitimate – as if it came from the intended DNS server. There are various tools available on the Internet to perform this attack, but Etthercap and Websploit are popular tools for performing DNS-based MITM attacks.

2.7.1.2 DNS ID Spoofing

DNS ID Spoofing is a kind of Man-in-the-Middle attack. The DNS query submitted by the stub resolver includes a unique identification number called QID of the DNS header section, used to link queries and responses. If the adversary can guess this QID, then the adversary will spoof the recursive resolvers IP address, making a message seem legitimate – as if it came from the intended DNS server. Therefore, DNS ID spoofing involves duplicating the QID and IP information generated for the resolve request submitted by the client and inserting false information. The stub resolver accepts the response containing the unexpected information since the response QID fits the request

QID. The unique identification numbers are generated by a pseudo-random number generator (PRNG) in DNS configuration software like BIND. Suppose a few consecutive IDs can be predicted. In that case, the entire PRNG function can be predicted, which is a limitation discovered in an older version of the BIND software [21].

2.7.1.3 DNS Cache Poisoning

The DNS cache poisoning attack is a form of DNS ID spoofing on the recursive resolver by the adversary to insert false DNS records inside the DNS cache of the recursive resolver to redirect users to the desired malicious websites controlled by the adversary [28][29]. Most DNS information is stored in the DNS cache for anywhere from an hour to a day after it is initially submitted. Although this is useful for reliability since a cache search takes considerably shorter time and effort than a DNS query to the authoritative server which follows DNS hierarchy again, it also ensures that if an adversary can supply the cache with deceptive yet legitimate data for a common address, they can easily redirect a significant number of users throughout the future. However, for the bogus answer to appear true, the adversary may need to do some things, such as spoofing the IP address of the actual authoritative DNS server queried by the recursive resolver, predicting the Query ID, and sending port used by recursive resolver [22]. The recursive resolver will then store the bogus answer in its DNS cache before the genuine one comes from the actual authoritative server. The genuine one will be discarded subsequently. This occurs because the recursive resolver only records and stores the first response to the question and ignores the others.

As shown in the Figure 2.13, the recursive resolver gets a DNS query from its one of the client for the domain "demodns.com", let us assume if DNS cache does not contain the record for the requested domain, then the recursive resolver follows the DNS hierarchy and redirects to forwards the query to one of the authoritative DNS servers for the requested domain. When the adversary notices the query, he creates a fake DNS response with the IP address he needs to poison inside the DNS cache of the recursive resolver. The source IP address in the IP-header of the false reply message must be the IP address of the authoritative DNS server, and the ID field in the false reply message must match the ID field in the question message. Since recursive resolver queries multiple DNS servers in the hierarchy for knowing the domain's mapping, the latency between sending the query and receiving the response is high. As the latency increases, the intruder will have more time to formulate a bogus answer to the requesting host.

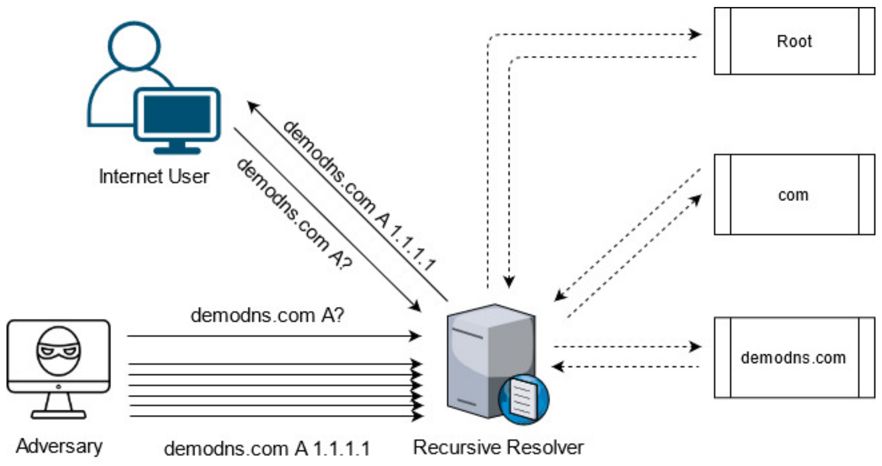


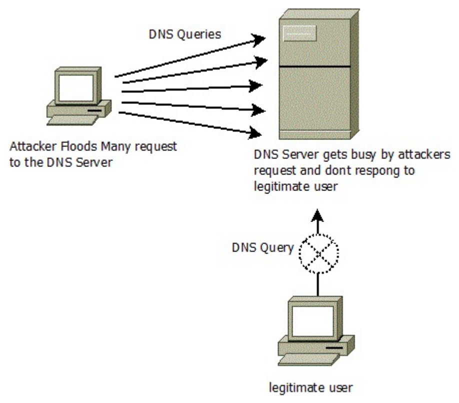
Figure 2.13 DNS Cache Poisoning Attack

2.7.1.4 NXDOMAIN

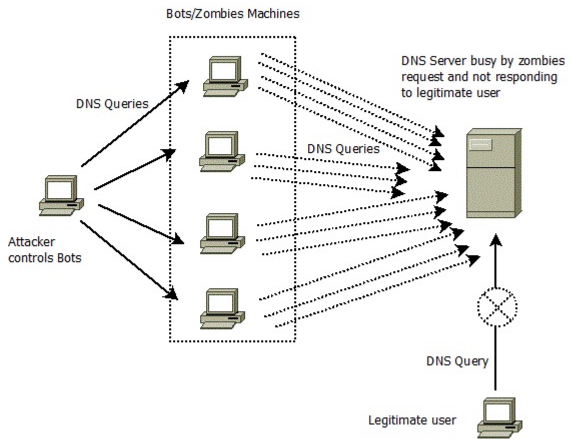
NXDOMAIN attacks, also known as water torture attacks or random subdomain attacks, can target both the authoritative DNS server and the recursive resolver. The adversary creates a DNS query with a non-existent subdomain or host of the domain name and sends it to the target recursive resolver. As in the process of resolution through iterative DNS query to the entire DNS hierarchy, the recursive resolver finds these domains are invalid or non-existence and reply with NXDOMAIN results. To achieve efficiency, the recursive resolver inserts the NXDOMAIN entry into its cache so that future requests for the same domains are answered from the DNS cache [23]. As a consequence of this operation, the resolver's cache is exhausted by random and useless replies, reducing the recursive resolver's performance and resilience. Suppose a large number of queries are created in a short period of time using Botnets. In that case, the cache rapidly fills up, and legitimate users experience significant delays in receiving responses, making this attack a more sophisticated distributed denial of service (DDoS) attack.

2.7.1.5 DNS Flood

DNS flood is a variant of DoS attack and can target any authoritative or recursive DNS server available for querying over the Internet. The DNS flooding attack aims to deplete server-side resources through a storm of UDP-based DNS queries from numerous malware-infected systems [24]. It is a UDP flood that targets DNS server port 53, as DNS normally uses UDP for query and response. There are two variants of DNS flood attacks. In the first variant, an adversary uses a single link to overload the target DNS server with bogus DNS queries to deplete the DNS server, as shown in Figure 2.14a. The aim is here to overload the DNS server with a huge amount of DNS queries that consume all its available resources like CPU memory and network, resulting in a slightly longer response time for genuine DNS queries. As shown in Figure 2.14b, the second variant uses a collection of compromised systems generally called BOTNET to create massive DNS requests over UDP, resulting in a disruption in targeted DNS Server availability [25][26].


DNS Queries
Attacker Floods Many request to the DNS Server
DNS Server gets busy by attackers request and dont respong to legitimate user
DNS Query
legitimate user

(a) DoS Attack



(b) DDoS Attack

Figure 2.14 Variants of DNS Flood Attack

2.7.1.6 Modified Data Attack

DNS modified data attack is built on the interception of shared data. A single DNS domain can be hosted on multiple authoritative DNS servers one among them is called as primary and rest all can be secondary. The authoritative DNS server can be configured either by command line interface (CLI) or graphical user interface (GUI). Normally, zone files or DNS zones can be configured by executing certain commands on the command line console, but they can also be configured with a graphical user interface. The Zone file may be changed or removed to affect the DNS query comprehension or avoid information from reaching the intended recipients [27]. The adversary must first obtain admin privileges by launching attacks against existing vulnerabilities in DNS server applications to carry out a modified data attack. The DNS zone data will then be modified remotely, leading users to malicious websites or communications.

2.7.1.7 Corrupted Data Attack

The DNS name resolution method can fail if an attacker may compromise an authoritative name server's zone file with false resource records or negative responses. This is a kind of modified data attack in which the attacker's goal is to corrupt the zone files, causing the domain to be unavailable to Internet users. For example, an incorrect NS or A record for "nitk.ac.in" zone will make the domain unreachable.

2.7.1.8 Spoofing Master Attack

DNS zones are mostly stored in the master or primary DNS server, with a copy being downloaded by the slave or secondary DNS server. The master name server is usually the only one on which data can be written. Moreover, a synchronization mechanism from the master name server called zone transfer can be initiated by the master or ordered by the slave name server. The adversary simply impersonates as a slave DNS server and requests a copy of the critical zone file from the master DNS server [33]. The zone file content may reveal a lot about the internal network's topology, which adversaries can use to footprinting that leads to major attacks [30].

2.7.1.9 Spoofed Updates Attack

Rather than manually updating DNS records by modifying complex zone files, a DNS client may use dynamic updates to change the domain name to IP address mappings in the name server. The dynamic update feature allows remote DNS clients to enroll and dynamically update their DNS records with the DNS server. This reduces the need for manual zone record administration, particularly for clients whose IP address changes frequently or uses DHCP protocol to gets an IP address. The dynamic updates are usually secured by providing access control list such that registered and authenticated clients can do dynamic updates. Spoofed updates attacks take advantage of this functionality by spoofing the IP address of registered clients and pushing fake records or deleting the DNS server's original records, which may sometimes lead to DNS server denial of service [31].

2.7.1.10 TCP SYN Flood Attack

If a DNS query fails over UDP, the message is sent over TCP port 53 instead of UDP. TCP is often used for zone transfer between master and slave servers. An adversary may take advantage of this by launching an SYN flood DoS attack, consuming all available server resources, and leaving the server inaccessible to normal users. The attack makes use of the TCP protocol's core features, the three-way handshake. The adversary sends many SYN packets to the targeted server by spoofing the IP address of identified slaves of the authoritative DNS server. After that, the server reacts to each request and keeps an available port open to accept the answer. The adversary keeps flooding new SYN packets as the server waits for the final ACK packet, which never arrives. Each incoming new SYN packet allows the server to open a new port for a set amount of time. After all of the available ports have been used, the server cannot normally run, resulting in a DoS attack [32].

2.7.2 Attacks Exploiting the DNS Infrastructure

Since security was not a priority when DNS was created, it was designed for usability instead. Over time, it grew in strength; DNS is now targeted by a sophisticated and complex range of attacks. DNS traffic flows freely across network perimeter and intranet network segments. Organizations cannot just obstruct UDP port 53 traffic because it will break most, if not all, network communication. Adversaries are well aware of this and have devised methods to use DNS to their advantage. There are a variety of strategies that use the DNS infrastructure to launch attacks on other Internet nodes.

2.7.2.1 DNS Reflection

The open recursive DNS servers are used to launch the DNS reflection attack. The intruder takes advantage of this by crafting a DNS query that spoofs the victim system's IP address and sends it to an open recursive server, which performs the resolution process and returns the response to the victim system that consumes network bandwidth at victim [34]. Since DNS uses UDP, the recursive resolver believes the query came from the target system and returns it. The DNS server always returns the response to a DNS query to the query's originator. The victim would misinterpret the answer as coming from an intermediary DNS server and will disregard it. When the victim receives an excessive number of duplicate DNS response packets, the time it takes to process and discard them grows exponentially, causing the network to become congested and the victim to go down, resulting in a denial of service attack. The prominent characteristic of a reflection attack is that the intruder remains undetectable.

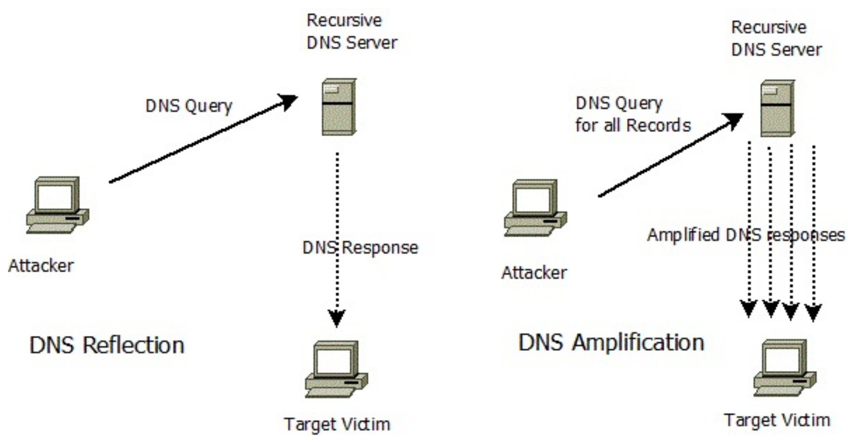


Figure 2.15 DNS Reflection and Amplification Attack

2.7.2.2 DNS Amplification

DNS amplification is a complex denial-of-service attack that uses the DNS server's behaviour to magnify the attack. The DNS amplification attack employs DNS reflection to overwhelm the victim with massive responses from a list of recursive resolvers, as shown in Figure 2.15. The adversary executes a variety of malicious activities in order to carry out the amplification attack. First, the adversary spoofs the victim's IP address and uses a recursive resolver to send DNS queries. This will send all of the recursive resolver's DNS responses to the victim's machine. Second, the adversary discovers an Internet domain with several DNS records. Because DNS queries that request the whole list of being requested records for a domain generate huge responses from the recursive resolver, they must normally be spread into many packets. Recursive resolvers respond with substantially bigger responses, perhaps up to a hundred times larger [35]. As a result, if the intruder produces two megabytes of DNS queries per second, the attack traffic on the victim is multiplied to two hundred megabytes per second, resulting in victims' infrastructure being affected by the unexpected load and may go down.

2.7.2.3 DNS Tunneling

Tunneling is one way in which adversaries manipulate DNS and exploit it to circumvent security. This attack allows for command and control (C2) and data exfiltration traffic that most enterprises do not monitor or are unable to detect. It uses basic DNS queries and answers to encode information of various programs or protocols [36]. To carry out this attack, the adversary must install malware on one of the systems in the targeted organization, which sends encoded DNS queries for a domain name owned and hosted by the adversary. The encoded DNS queries are sent to the organization's legitimate local DNS server, which forwards them to the requested domain name's authoritative DNS server. Since this authoritative DNS server is in the adversary's control, it decodes all these queries. It creates a tunnel among the victim system and the authoritative DNS server by encoding DNS queries and responses. This tunnel can be used to steal an organization's confidential information, i.e., data exfiltration to the system of the adversary, or send basic commands to a remote access trojan to enable a C2 channel, or even total IP traffic can also be tunneled [37].

2.7.2.4 DNS Hijacking

During an Internet session, DNS hijacking manipulates the transaction and leaves users unaware of the DNS servers they are using. It is a malicious exploit in which users are routed using a rogue DNS domain, which switches the redirected Internet user's IP address [38]. Either downloading malware carries out the DNS attack on the victim's device or hacking DNS communication by man in the middle attack. DNS hijacking, also referred to as DNS redirection, affects DNS clients infected with malware or Trojans. It modifies the client's TCP/IP settings to direct DNS requests to a rogue DNS server controlled by the attacker [39], the rogue DNS server will then redirect web traffic to malicious websites.

2.8 Literature Survey

2.8.1 Literature on DNS Health Measurement

Typical information and communication technology - ICT architectures of most critical infrastructure like energy sectors, communication sector, financial service sectors, nuclear reactors, materials and waste sector, transportation system sectors etc., adopt traditional ICT networks for interconnecting their sub-components. Because DNS is a crucial component of ICT networks, it is classified as a critical infrastructure [40]. With this realization, several technological improvements have been made to defend against DNS exploitation. As a result, the overall health of the DNS infrastructure is critical to the Internet ecosystem. DNS tracking, diagnostics, and wellbeing have significant implications for today's Internet and are primarily investigated by Internet researchers and engineers.

In February 2009, the Georgia Institute of Technology hosted the "1st Annual Global Symposium on DNS Security, Stability, and Resiliency", intending to bring together cross-functional stakeholders to highlight DNS threats and related issues. The outcome of the symposium led to the identification of the major challenges, existing initiatives, and potential solutions to difficulties facing the DNS and the validation of the necessity for and advantages of continuous collaboration across fields [41].

The 2nd global annual conference on DNS security, stability, and resiliency was held in 2010 by ICANN to assess DNS health [42]. The conference produced a definition of DNS health and five primary measuring indicators for evaluating DNS health: coherency, integrity, speed, availability, and resiliency. Infrastructure operators, DNS operators, policymakers, and end-users were all mentioned as four distinct points of view on the notion of health. ICANN has identified three main areas of potential work that it wants to address, i.e., Identifying DNS security threats, designing activities to assess the DNS resiliency and improve its operating processes, and identifying methods for establishing a DNS health measurement.

Several studies on DNS health have been done in the past, all of which claim to evaluate DNS health. However, there are still several unresolved issues for DNS health measurement, including a lack of consensus on determining health indicators, a common concept of typical DNS behavior, and a lack of a standard framework for data or information exchange. The authors of [43] suggest combining several DNS health and security metrics in aggregated indexes.

The Measuring Naming System (MeNSa), proposed by the authors in [44] [45], is a framework for assessing DNS health that includes a formal methodology, metrics, and tools. The authors also provide a method for combining health and security metrics to generate threat indicators. The framework and aggregate metrics are useful, as evidenced by the results of a scenario-based experiment in their research work. The MeNSa project proposed a systematic and standardized methodology for assessing DNS protection and health. The author added two more indicators to the five key indicators, bringing the total number of indicators for determining DNS health to seven; stability and vulnerability are the two new metrics. Their work on the MeNSa platform, on the other hand, is limited to vulnerability, resiliency, and security indicators for assessing DNS health. We expanded on this work and defined three major parameter classes (vulnerability, DNS latency comparison with ICMP latency, and DNSSEC validation) for assessing the authoritative name server's health concerning all prescribed indicators. The MeNSa Project aims to define a series of metrics and an algorithm to measure the DNS health level of various functions and from various perspectives.

Tejaswini et al. [46] identified some critical nodes as root DNS and few TLD DNS. They suggested a simple method for estimating health by periodically monitoring these identified nodes in the DNS hierarchy. They used DNS probing method to find out the response time, which is used to measure health. The average response time was estimated upon gathering the information over a period of time, and thresholds was determined that might be used as a cut-off for assessing if the critical node was in a critical state. For evaluation of DNS health, the authors only used one parameter: query response time which belongs to the speed indicator of ICANN and MeNSa.

By executing passive probes and detecting the state based on previous behaviours, C.Yamini et al. [47] present a simple but worldwide DNS network visualisation technique. They identified critical nodes and developed tools for probing and monitoring their condition over time.

In their research paper titled " DNS Recursive Server Health Evaluation Model ", Zhaoxin et al. [48] presented a technique to evaluate the health of recursive DNS servers. They use three key indicators for evaluating the health of recursive DNS: availability, integrity, and confidentiality. The identified parameters include DNSSEC, EDNS0, software version type, and software vulnerabilities, corresponding to the three key indicators. The proposed method was experimented against 13512 recursive DNS servers of Jiangsu province of China.

Casey Deccio [49] addressed the difficulties of diagnosing and measuring DNS systems and suggested a tool called looking glass for DNS measurement and diagnostics that is lightweight, flexible, and easy to implement. On the other hand, the implementation only accepts DNS queries from organizers/analysts via HTTP/SSH protocols and sends them to the specified server on behalf of the organizer.

By studying the consistency of the DNS server on the Indonesian island of Sumatera, Rizal Munadi et al. [50] investigated the health of DNS. They considered only two parameters, i.e., response time and reliability, to assess the system's health. The DNS benchmark and visual route on the Linux machine were used to assess the system's health. In this study, ten different institutions were chosen, and two-parameter evaluations were obtained and tabulated. DNS servers in universities were analyzed based on DNS health and classified into healthy and unhealthy categories.

For evaluating the health of the authoritative DNS servers, Jian J et al. [51] used four metrics, i.e., DNSSEC, redundancy, recursion, and query latency. Their research focused on the analytic hierarchy method (AHP) for DNS health measurement and defined new metrics and parameters. For measuring the DNS, health authors used six Points of View (PoVs) defined by the GCSEC, i.e., end-user, application service provider, resolver, name server, zone, and global.

Several research studies [52 -54] relate to DNS traffic monitoring and performance metrics, but few report on DNS health measurement.

2.8.1.1 Outcome of Literature Survey for DNS Health Measurement

DNS health is an important aspect of the DNS infrastructure; the identified indicators for DNS health measurements are:

- Availability: The DNS service should be available and accessible when a DNS query is sent by the DNS client.
- Coherency: One of the fundamental concepts of DNS is the capacity of DNS to correctly maps the domain name to the IP address and vice versa.
- Integrity: DNS's ability to protect against unauthorized data manipulation or loss, including database non-repudiation and authenticity.
- Resiliency: In the event of a failure, the DNS servers have the potential to revert to their original state.
- Speed: The efficiency of DNS concerning response time and throughput. In addition to DNS queries, speed applies to maintenance, administration, and management operations.
- Stability: The capacity of DNS to perform in a consistent and predictable manner (e.g., protocols and standards). Stability is critical since it allows for wide acceptance and utilisation.
- Vulnerability: The probability that a DNS flaw would be abused by one or more attacks.

While there are numerous studies on DNS measurements and performance assessment, only a few on DNS health measurement regarding security, stability, and reliability, meanwhile, the researchers of this domain must develop a high-level consensus on DNS vital signs that can be used to determine system health, including coherency, integrity, speed, availability, and resiliency. DNSTOP, DNS statistics collector, DNSViz, and others have recently been developed to visualize DNS node status, trace and help to limit domain names used for malicious purposes. Alternative DNS Health Checkers include Pingdom DNS Check, whatmydns.net, MXtoolbox, DNSStuff, dns.squish.net, and others [55-60]. Measuring the Naming System (MeNSa) presents a systematic & standardized technique and a collection of metrics for assessing DNS health and security.

2.8.1.2 Identified Research Gaps from the Literature Survey for DNS Health Measurement

There are a number of initiatives in this domain, all of which claim to monitor DNS health from a local viewpoint. The reality is a little different, and several obstacles remain: there is no standard measure (just a shared list of five health indicators); no agreed-upon method for computing health indicators; and no agreed-upon definition of normal DNS behaviour. Even though many methods to measure DNS health have been presented in the literature, none account for all of the health metrics identified by ICANN and added by the MeNSa system. This thesis bridges the difference by considering all health indicators and dynamically measuring DNS health by proposing a new approach and new criteria for assessing DNS health.

2.8.2 Literature on DNS Intrusion Detection

Anna Drozdova's thesis work [61] focused on developing and testing a system to safeguard DNS servers in a lab environment by installing an IDS at a key position - proximity to the DNS server - and implementing some snort rules to block malicious website lookups. The test lab was set up in a LAN environment with three systems: a victim system, an attacker system, and a SNORT IDS system. The experiment was carried out in a secure LAN environment. The SNORT system's infrastructure is being investigated for DNS security, although DNS server protection is neither complete nor comprehensive. However, this work aims to demonstrate the behavior of the DNS server protection system in a real network and investigate the capabilities of the SNORT system and its configuration settings for DNS server security.

In [62], Filip Hock and Peter Kortis proposed the notion of "Design, Implementation, and Monitoring of the Firewall System for DNS Server Protection," which deals with

DNS security techniques applicable on the transport and network layers. Traffic shaping, flow filtering, and priority are used in the suggested protection mechanism. They demonstrated how a firewall might alter traffic dynamically to ensure that packets arrive while the DNS server is under attack. They stated the guidelines for developing traffic shaping and prioritization rules.

Bong-Hyun Kim [63] created the client control system using DNS dependability of the system control block for changing and diversified security breaches in advance for block module in his work "Design and Analysis of Client Control System Using DNS Control Firewall." Dynamic IPS module design, embedded DNS system module design, Interlocking DNS service module design, and Cert & Analysis module design were all part of the client control system design. The DNS control firewall is commonly used in the design of client control systems.

Joao Afonso et al. [64] described a system that uses network sensors to perform real-time network monitoring to identify, block, or limit the extent of an attempted intrusion into the DNS application. The proposed solution employs heuristics to identify traffic is harmful to the DNS by evaluating all data collected from various sources in real-time and assigning appropriate weights to each component (occurrences, analysis, time between occurrences, incidence, and intrusion detection system) acting appropriately.

Pratick Satam et al. [65][66] describe an anomaly-based IDS that runs in two phases, training and operational, to identify aberrant behavior and exploitation for DNS protocol. In the training phase, the usual behavior of the DNS protocol is simulated as a finite state machine, with temporal statistics kept in a database. In a different database, an irregular DNS traffic transition is noticed. An anomalous metric is created based on these two statistics, and it is utilized in the operational phase to identify various DNS assaults. The study was carried out using various known DNS assaults. However, they did not examine DoS/DDoS assaults on DNS during the process.

Steven Cheung et al. [67] proposed a detection-response-based method for protecting DNS against different threats, including DNS cache poisoning and spoofing. It is used in conjunction with an IDS strategy and is driven by formal specifications that enable reasoning. The authors created a DNS wrapper that monitors all incoming and outgoing

DNS messages from the DNS server for security violations. DNS traffic is examined in comparison to their standards, and any variations are flagged as unusual. This event is identified as a potential attack if the observed traffic differs from the authoritative answer.
In [68], the authors presented a machine learning-based IDS for DNS DoS assaults, which aims to recognize DNS DoS attacks using the learning capacity of neural networks. Back Propagation, Radial Basis Function, and Self-Organizing Maps neural networks were used to test the suggested IDS. On the ns-2 simulator, the complete experiment is simulated. The findings show that a BP neural network outperforms other types, with a solution accuracy of 99 percent and a low false alarm rate for attack detection.

2.8.2.1 Outcome of Literature Survey for DNS Intrusion Detection

The DNS protocol is based on UDP, a connectionless data exchange protocol. Any hacker can easily spoof a legitimate IP address to generate attacks on DNS services.
Much research has been done to secure DNS from tunneling attacks, amplification attacks, and DNS DoS attacks. However, there has been no study done to defend DNS from all potential attacks. Further more from the literature survey, it is being observed that traditional security solutions such as firewalls, IPS, or IDS are not purpose-built DNS security systems, and have no in-depth understanding of the DNS protocol, and are not adapted to protect DNS infrastructure, including host, root, TLD, SLD, and RR.

2.8.2.2 Identified Research Gaps from the Literature Survey for DID

The DNS infrastructure must work properly in order for the Internet to be available and accessible. It is consequently critical to safeguarding the architecture on which it is built and the records it contains. DNSSEC implementation at all domains in the hierarchy can handle most key risks such as MITM and cache Poisoning; nevertheless, DNSSEC cannot defend against all attacks. As the inbuilt signature are not able to identify all DNS threats, standard security technologies, such as firewalls, intrusion prevention systems, and intrusion detection systems, are not designed and adapted to protect DNS infrastructure, including root, TLD, SLD, RR, and DNS client. They lack an in-depth understanding of the DNS protocol. This necessitates creating a DNS Intrusion Detection (DID) system that guards against all types of DNS assaults by defining appropriate attack signatures for all attack surfaces.

2.8.3 Literature on DNS Hierarchy Testbed Setup and its Evaluation

LDplayer [100], a DNS testbed created by Liang Zhu et al. in 2017, supports several distinct levels of DNS hierarchy and numerous domains. This testbed was built to conduct large-scale experiments to determine how traffic volume varies when all DNS queries use the DNSSEC protocol and how server memory and network latency vary when all queries are sent over TCP rather than UDP. The testbed efficiently simulated the whole DNS hierarchy in a controlled environment where replays do not leak traffic on the real Internet, and tests may be repeated. In order to resolve DNS, the testbed supports both UDP and TCP transport layer protocols. However, the testbed was only established over IPv4 protocol.

The Yeti DNS Study [101] is an effort to create a live root DNS server system testbed for advanced root services, including certain IPv6-only operations, DNSSEC key rollover, renumbering issues, scalability difficulties, and other challenges. The objective of this alternative root system is to figure out the boundaries of DNS root name services and provide relevant technical information. The initiative, funded by a Chinese government organization, aims to test various DNS-related new technology to enable sovereign countries to investigate and regulate the Internet and strengthen their network sovereignty.

In [102], Soohong Park et al. presented four deployment scenarios for setting up DNS in IPv6 wired and wireless networks and offered instructions to setup DNS for IPv6 network managers and users to utilize in their target networks. The authors additionally looked at three IETF-proposed techniques for DNS setup in IPv6 hosts, including recursive DNS server addresses and the DNS search list. According to the study, IPv6 equipped clients can get DNS addresses via router advertising DNS choices, DHCP DNS possibilities, well-known DNS anycast addresses, and DNS query messages. Researchers did not, however, discuss how to establish DNS on a dual-stack system.

Fuliang Li et al. [103] tested IPv6 network performance in form of network reachability, packet reordering and packet delay/loss. A comparison study was conducted on IPv6 packet delay, IPv6 packet-loss by a probing tool, OneProbe. The paper concludes stating, IPv6 and IPv4 have similar packet delay and loss. But through our experiment we arrived at a different result in case of DNS queries.

In [104], the performance of UDP over IPv6 was tested for several versions of Windows (Windows 7 and Windows Server 2008) and Linux (Ubuntu 10.04 and RedHat 5.5), and the experiment was done using a peer-to-peer connection between client and server computers. The authors utilized various traffic-measurement software, including IP traffic for Windows and Iperf for Linux systems. According to the authors, the UDP throughput in IPv4 is greater than IPv6.

In [105], authors emphasized several measuring techniques in their research article and concluded that IPv6 networks had a greater latency and loss than IPv4 equivalents. In [106], the author used the Iperf network measurement application to evaluate the network throughput of IPv4 and IPv6 on Linux systems, noting network performance parameters including latency, throughput, and jitter for both protocols. They conclude that when utilizing smaller packet sizes, the IPv6 header cost starts to pile up.

N Shanel et al. [80] compared the performance of IPv4 and IPv6 on six OS, including major Windows and Linux variants. According to the findings, network performance is influenced by IP version and traffic type and by the OS selected.

The research was done in [82] to revisit the performance of DNS latency for several public recursive resolvers, including Google DNS, Open DNS, Cloudflare, and Quad9. They discovered that IPv6 has greater DNS query latency than IPv4 for specific resolvers after analyzing the influence of IP version choice on DNS latency.

2.8.3.1 Outcome of Literature Survey for DNS Hierarchy Testbed Setup and its Evaluation

Several studies comparing IPv6 versus IPv4 network performance in terms of TCP and UDP protocols have been conducted. Moreover, very few study has been done to compare DNS performance with network-layer protocols. The IPv4 subsequent queries can take a different route in the real DNS hierarchy, and the same is true for the IPv6 network. Therefore the DNS query latency in the real DNS hierarchy cannot be compared directly from a particular vantage point due to variations in the number of hops of query on IPv4 and IPv6 communication networks. This research work aims to undertake the first-ever DNS query latency comparison on the DNS hierarchy testbed for IPv4 and IPv6 protocols.

2.8.3.2 Identified Research Gaps from the Literature Survey for DNS Hierarchy Testbed Setup and its Evaluation

1. The guidance required for commissioning DNS hierarchy testbed setup over IPv6 is widely lacking.
2. None of the research brought up the operational difficulties that arose throughout the deployment and provisioning of services.
3. None of the research gives a clear illustration and guidelines for setting up at least three level of DNS hierarchy (ROOT, TLD, STD, Subdomains, and Recursive Resolver) on top of IPv6, enabling forward and reverse lookup tree.
4. A lot of research work has been conducted on IPv6 performance, but none has been done on DNS query latency compared to network-layer protocols.

Due to differences in the number of queries hops on IPv4 and IPv6 communication networks, the DNS query latency from an Internet vantage point for IPv4 and IPv6 networks cannot be compared directly. Furthermore, there is no guarantee that the DNS server in the hierarchy is dual-stack hosted.

2.9 Summary

We introduce some key concepts and technologies related to the DNS ecosystem. The background of DNS is presented, as well as the protocol structure and various threats related to DNS. In addition, we also conducted a comprehensive literature review on the DNS health measurement, DNS intrusion detection, and DNS hierarchy testbed setup with DNS query latency evaluation of IPv4 vs IPv6. We also highlighted the research gaps for those issues to support the research requirements.

Chapter 3

DNS Health Measurement

3.1 Introduction

The authoritative name servers, recursive resolvers, and root DNS servers makes up the DNS infrastructure. To ensure smooth and efficient query resolution, every part of the DNS system must function properly. There are approximately 367.3 million registered domain names as of the end of 2020 [77], and thousands of authoritative name servers serves these domains. The DNS system relies heavily on authoritative servers as all information in the DNS for a domain is served by authoritative servers for that domain. In order to function properly, Internet networks depend heavily on DNS response time and accuracy. Since authoritative DNS servers are susceptible to various attacks such as DoS/DDoS, spoofing master, spoofed updates, and corrupted data/modified data, the proper level of DNS health for a resilient and robust Internet is a major challenge, and its health determines the DNS system's performance. The overall DNS server health must be understood in order to anticipate large-scale attacks and take precautionary steps. Determining the health of a large-scale distributed infrastructure such as DNS, on the other hand, is difficult, particularly without intruding into one's network. Since DNS spans the entire Internet and contains millions of nodes, evaluating its overall health is a huge challenge requiring the installation and configuration of millions of probes, which is not feasible due to various practical constraints, including security threats. We suggest that such a distributed and global system can be assessed by examining the health of a few crucial nodes like authoritative name servers, i.e., nodes with the ability to invigorate or thwart the entire DNS system. Since authoritative DNS servers serve all of the information in the DNS for a specific domain, their health is directly related to the integrity, consistency, reliability, and accuracy of the resolved data. As a result, determining the health status of authoritative DNS servers has become a pressing issue in network security today.

We suggest a new approach for assessing the health of authoritative name servers, which are a vital, central, and significant part of the DNS ecosystem. For most Internet components, efficiency is usually calculated in response time, reliability, and throughput. DNS software weaknesses, DNS latency comparison with ICMP latency, and DNSSEC validation are among the parameters class we suggest for evaluating the health of authoritative name servers. The identified parameters help figure out how authoritative name servers behave in general, detect sluggishness in their results, and calculate a health score. We evaluated the health of authoritative servers for the world's top 500 domains by probing algorithms to evaluate these parameters and computing the overall score. This method should be used regularly to examine the circumstances and take the appropriate steps to secure authoritative domain name servers from attacks.

3.2 Methodology and Proposed Health Parameters

Misconfigurations, vulnerabilities, and malicious activity may all influence DNS performance, affecting the entire Internet ecosystem. As a result, it is critical to track, analyze, measure, and diagnose the DNS ecosystem's health. We choose the probing approach for DNS health since it is a straightforward DNS measurement and diagnostics method. We came up with eight major parameters for evaluating the health of authoritative DNS servers: software obfuscation, SOA validation, dual-stack support, non-availability of zone transfer status, recursion validation, reverse lookup validation, DNS latency comparison with ICMP latency, and DNSSEC validation. We considered all of the main indicators mentioned by ICANN and MeNSa, i.e., availability, coherency, integrity, resiliency, speed, stability, and vulnerability. Figure 3.1 shows identified parameters with respect to main indicators.

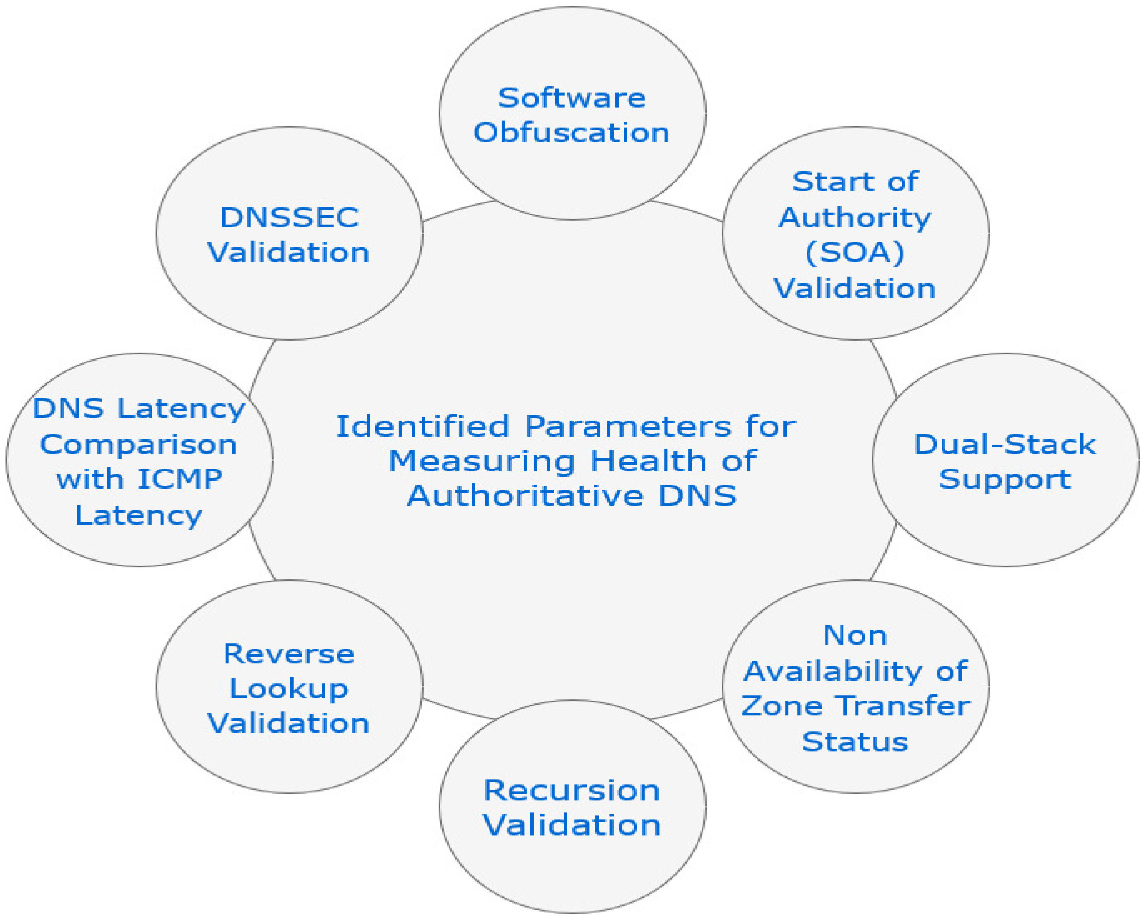

Software Obfuscation
Start of Authority (SOA) Validation
DNSSEC Validation
DNS Latency Comparison with ICMP Latency
Identified Parameters for Measuring Health of Authoritative DNS
Dual-Stack Support
Reverse Lookup Validation
Non Availability of Zone Transfer Status
Recursion Validation

Figure 3.1 Identified Parameters for Health Measurement of Authoritative Name Server

3.2.1 DNS Vulnerabilities

In order to thwart communications or direct unwitting end-users to fake web pages or other destinations, attackers may target DNS services directly. On the other hand, DNS could be used as a mediator in a larger network attack. DNS protocol does not provide sufficient authentication capabilities to prevent unauthorized read/update of configuration files, domain zone files, and files containing signing keys files [79]. Misconfiguration of DNS software is the most common cause of vulnerabilities, and incorrect DNS service and zone information configuration may result in incorrect resolution or server behavior. We identified the following parameters for the DNS vulnerabilities checks: software obfuscation, SOA validation, dual-stack support, non-availability of zone transfer status, recursion validation, and reverse lookup validation. We propose a vulnerability check to identify the misconfiguration and flaws in the DNS server. The suggested parameters for calculating the vulnerability score are shown in Figure 3.2.

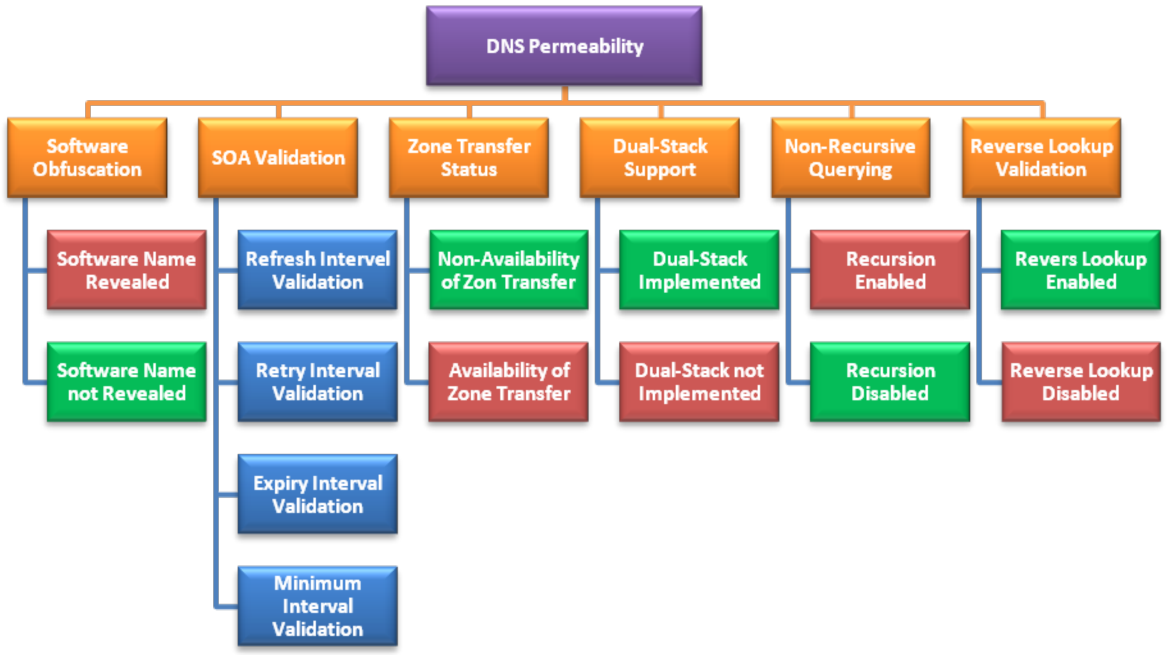

DNS Permeability
Software Obfuscation
SOA Validation
Zone Transfer Status
Dual-Stack Support
Non-Recursive Querying
Reverse Lookup Validation
Software Name Revealed
Software Name not Revealed
Refresh Intervel Validation
Retry Interval Validation
Expiry Interval Validation
Minimum Interval Validation
Non-Availability of Zon Transfer
Availability of Zone Transfer
Dual-Stack Implemented
Dual-Stack not Implemented
Recursion Enabled
Recursion Disabled
Revers Lookup Enabled
Reverse Lookup Disabled

Figure 3.2 DNS Vulnerabilities Parameters

3.2.1.1 Software Obfuscation

Attackers can profile a DNS server by learning the names of its software. Information disclosure vulnerabilities like software name lookups are frequently neglected. It is preferable if the DNS client is unaware of the DNS server's software name. A new DNS software vulnerability is found now and then, and attackers scour the Internet for unpatched computers to exploit. As a best practice, hide software names on DNS servers; while this is not natural protection, it does make scanning servers a bit more difficult. The DNS server becomes subject to other attacks if an attacker learns the DNS software name. The DNS application must reject requests for the name of DNS software.

Each successive version of the DNS server software is generally free of vulnerabilities found in earlier versions since design improvements have been incorporated to mitigate specific issues.In some cases, upgrading to the most recent version of name server software may not be possible straight away. Adversary takes advantage of this and uses a variety of tools available on the Internet to determine a DNS server's name and version, which is the first step in attacking the DNS infrastructure, which malicious actors may then target in order to gain access to the network, from which they can further enter the network or use the DNS server for malicious purposes.

Because typical DNS misconfigurations influence DNS availability [78], this parameter is linked to ICANN's availability and MeNSa's vulnerability indicators. As a result, we use the DNS server fingerprinting tool - "fpdns" [81] for determining the DNS software name to see if the DNS server provides its software name.

We propose this software name check by Algorithm-1, which checks software names for all the authoritative DNS servers of a domain. SVCOUNT and NSCOUNT are the

Algorithm 1: DNS Server Software Name Check

```
Input  : Name of the domain - DOMAIN
Output: 0 (Test Failed) or 1 (Test Passed)
1  Initialize SVCOUNT, NSCOUNT to 0
2  SERVERLIST=List of all authoritative DNS servers of DOMAIN
3  for I=1 to number of authoritative DNS Server of DOMAIN do
4      NSCOUNT = NSCOUNT + 1
5      IPADDR=Get IP address of DNS Server in SERVERLIST[I]
6      MSG=output of "fpdns IPADDR" command
7      if MSG contains "No match found" then
8          SVCOUNT = SVCOUNT + 1
9      end
10     else if MSG contains "TIMEOUT" then
11         SVCOUNT = SVCOUNT + 1
12     end
13     if NSCOUNT==SVCOUNT then
14         return 1
15     end
16     else
17         return 0
18     end
19 end
```

counters for the number of authoritative name servers that do not reveal the software name for a given domain and counter for the number of authoritative name servers for a given domain, respectively. The SERVERLIST obtains a list of all authoritative name servers for the domain by following command: host -t NS DOMAIN. The IP address (IPADDR) of each authoritative name server is received, and the "fpdns" command is run on that IP address; the result is saved in the MSG variable. The "fpdns" command returns either the DNS software name or "No match found." In rare circumstances, firewalls will block communication for "fpdns," resulting in a "TIMEOUT." We look for the words "NO match found" and "TIMEOUT" in the MSG and use that to see if the authoritative name server exposes the software name or not. The domain passes this test if the software version is hidden on all authoritative name servers.

3.2.1.2 SOA Validation

Every domain must include an SOA record that identifies the most reliable DNS server for the specific domain. The domain administrator's email address, the domain serial number, and a list of zone refresh times are all contained in this resource record. It also provides information between primary and secondary DNS servers. The following SOA interval values are compared to RFC 1912 [107]:

Refresh Interval: The refresh interval specifies how long secondary servers must wait before transferring zones from the primary server. For zones that are updated often, this number should be low (20 minutes to 2 hours), and for domains that are modified on a regular basis, a higher value can be substituted (2 to 12 hours). As a result, we check these values as RFC 1912 proposes, a 20-minute (1200-seconds) to 12-hour (4320-seconds) refresh interval range.

Retry Interval: The time a secondary server must wait until seeking an update from an inaccessible primary nameserver is considered the retry interval. This interval must be a minor portion of the refresh interval. If the refresh value is in the range mentioned earlier (a 20-minute (1200-seconds) to 12-hour (43200-seconds) refresh interval range.), the retry interval is 5 minutes (300 seconds) to 2 hours (7200 seconds).

Expire Interval: The expire value indicates how long the zone information in the secondary server should be considered valid if a secondary server cannot reach the primary server to update even after retry intervals. The expire interval should be between 2 and 4 weeks, according to RFC1912.

Minimum Interval: This interval is the default TTL duration for resource records to

Algorithm 2: SOA Resource Record Check

```
Input  : Name of the domain - DOMAIN
Output: 0 (Test Failed) or 1 (Test Passed)
MSG=SOA lookup for DOMAIN
REFRESH=get refresh interval value from MSG
RETRY=get retry interval value from MSG
EXPIRE=get Expire interval value from MSG
MINIMUM=get minimum interval value from MSG
if REFRESH <=43200 and REFRESH >= 1200 then
    R=1
end
else
    R=0
end
if RETRY <= 7200 and RETRY >= 300 then
    RE=1
end
else
    RE=0
end
if EXPIRE <= 2419200 and EXPIRE >= 1209600 then
    E=1
end
else
    E=0
end
if MINIMUM <= 432000 then
    M=1
end
else
    M=0
end
P=R+RE+E+M
if P <4 then
    return 0
end
else
    return 1
end
```

stay in the cache of other nameservers. The minimum interval should be shorter than

five days, according to RFC1912.
Suppose these intervals hold values outside the defined range of RFC1912, and if Refresh and Retry Intervals are configured too little, the primary/master and secondary/slave servers may experience higher processing and network burden, resulting in the termination of Internet service. In another case, when Expire interval is too large, the secondary servers will offer inaccurate data to clients.
If the Refresh and Retry values in the SOA resource record of the master authoritative server are set very high, and the zone file is changed frequently resulting data mismatch between the master and slave authoritativ servers. This phenomenon is called a zone drift, a mistake that causes inaccurate zone data maintained by secondary name servers. The secondary server will often initiate zone transfers if the Refresh and Retry values in the SOA RR are set very low; this is called zone thrash, a mistake that increases the burden on both the master and slave authoritative servers. DoS may occur as a result of inaccurate data or increasing demand. As a result, the SOA parameter is linked to indices of availability and resiliency indicators of ICANN.
SOA validation is performed through Algorithm-2. The command "host -t SOA $DOMAIN" is used to acquire SOA values for the variable MSG. The MSG is used to get the REFRESH, RETRY, EXPIRE, and MINIMUM values. We look for these intervals in the RFC 1912 range; for example, REFRESH should last 20 minutes to 12 hours, RETRY should last 5 minutes to 2 hours, EXPIRE should last 2 to 4 weeks, and MINIMUM should be less than five days.

3.2.1.3 Dual-Stack Support

Dual stack refers to a device's ability to run both IPv4 and IPv6 at the same time, enabling hosts to access both IPv4 and IPv6 networks, making it a very flexible coexistence technique. It is the most straightforward technique of obtaining IPv4 and IPv6 addresses. Every networking device in a network, including switches that utilize IPv4 or IPv6, will be set up to run both IPv4 and IPv6 at the same time. For IPv4 communication, the IPv4 protocol stack is often used, whereas, for IPv6 communication, the IPv6 protocol stack is commonly used. DNS decides whether IPv4 or IPv6 is used; nevertheless, the IPv6 protocol stack prioritizes over IPv4. If at all practicable, a DNS server should be hosted on both an IPv4 and IPv6 network.

Consequently, the DNS server can be accessed through the IPv6 Internet even if the IPv4 interface is unavailable and vice versa. Because IPv6 has an inbuilt security mechanism called IPSEC, DNS over IPv6 is more secure than IPv4. This test determines if the authoritative DNS server is connected to a dual-stack network. The DNS resolution method for that client will take longer if the DNS server does not support dual-stack and the client is dual-stack enabled.

Because the dual-stack is a direct technique for enabling high performance, this check links to ICANN's availability and stability metrics. We developed Algorithm-3 for the dual-stack implementation check, which takes the domain name as an input and determines whether the dual-stack is implemented on all the authoritative name servers of that domain. NSCOUNT is a counter for recording the number of authoritative name

Algorithm 3: Dual-Stack Implementation Check

```
Input  : Name of the domain - DOMAIN
Output: 0 (Test Failed) or 1 (Test Passed)
1  Initialize NSCOUNT, DSCOUNT to 0
2  SERVERLIST= List of all DNS servers of DOMAIN
3  for I=1 to number of authoritative DNS Server of DOMAIN do
4      Increment NSCOUNT by 1
5      IPv4_Record= lookup of A record at DNS server in the SERVERLIST[I]
6      IPv6_Record=lookup of AAAA record at DNS server in the
         SERVERLIST[I]
7      if IPv4_Record or IPv6_Record contains "record" then
8          continue for loop
9      end
10     else
11         Increment DSCOUNT by 1
12     end
13 end
14 if NSCOUNT=DSCOUNT then
15     return 1
16 end
17 else
18     return 0
19 end
```

servers for a particular domain, whereas DSCOUNT is a counter for storing the number of dual-stack enabled authoritative name servers. IPv4 and IPv6 lookups (using the host command) are stored in IPv4_Record and IPv6_Record, respectively, for each authoritative name server. If both the IPv4 and IPv6 lookups fail, the host command returns "has no A record" and "has no AAAA record," respectively. Therefore if the "record" is present in any of the variables IPv4_Record or IPv6_Record, the authoritative name server is not dual-stack enabled. The algorithm returns "Test Passed" if all authoritative name servers have dual-stack enabled.

3.2.1.4 Non-Availability of Zone Transfer Status

The two categories of authoritative DNS servers are master and slave DNS servers; the master authoritative DNS server keeps track of the domain's information in the zone file at all times, and any changes to the zone file are communicated to all slave servers for that domain. From a security standpoint, DNS zone transfers give a wealth of reconnaissance data, and an adversary can use this information to map the network and conduct an attack.

Most system administrators like to give each server in their networks a descriptive name [69], such as MainDC, AddDC, HRMS, WSUS, etc. This saves the attacker much time figuring out which internal/external application server to attack. It is quite straightforward to generate a map of the whole zone if the adversary can induce and capture a complete zone transfer. An adversary uses an AXFR request in a DNS query to transport complete zone information. It puts much demand on network resources and bandwidth compared to conventional DNS queries. If the adversary performs this regularly and uses Botnet, the authoritative DNS server may become overburdened, resulting in Denial of Service to legitimate users. As a result, TSIG and ACL are suggested for secure and approved zone transfers. Unauthenticated, remote users can take advantage of probable AXFR zone transfers to obtain important reconnaissance data used in further attacks. As a result, MeNSa's vulnerability indicator is linked to the zone transfer check parameter.

We propose Algorithm-4 for validating the zone transfer for all authoritative name servers of a domain in question. If the dig command is used to transfer a zone, the

Algorithm 4: Zone Transfer Check

```
Input  : Name of the domain - DOMAIN
Output: 0 (Test Failed) or 1 (Test Passed)
1  Initialize NSCOUNT, ZTCOUNT to 0
2  SERVERLIST= List of all DNS servers of DOMAIN
3  for I=1 to number of authoritative DNS Server of DOMAIN do
4      NSCOUNT = NSCOUNT + 1
5      MSG=AXFR lookup at next DNS server in the SERVERLIST[I]
6      if MSG does not contains "XFR size" then
7          ZTCOUNT = ZTCOUNT + 1
8      end
9  end
10 if NSCOUNT=ZTCOUNT then
11     return 1
12 end
13 else
14     return 0
15 end
```

entire zone will finish with the XFR size (which lists the number of records, messages, and bytes transferred). As a result, we examine the MSG variable to determine if it includes "XFR size." If "XFR size" is not included in MSG, the zone will not be downloaded, implying that the authoritative name server is secure for zone transfer.

3.2.1.5 Recursion Validation

At the same time, a DNS server might be authoritative, recursive, or both. Unless it is an internal server or serves name requests on purpose, the recursive option should be disabled. However, allowing recursive DNS requests on authoritative DNS servers poses a security concern since an adversary can conduct DNS reflection attacks. This attack may also be modified to magnify the response, resulting in an amplification attack in which the authoritative server launches the DoS attack itself. The main function of an authoritative name server is to resolve domain names for the zones for which it holds authoritative data. As a result, the recursion needs to be disabled for an authoritative DNS server, which prevents its use as a reflector for DDoS attacks. Therefore, blocking recursion on an authoritative name server protects the DNS server against reflected amplification DoS/DDoS attacks. This parameter matches the resiliency (capacity to withstand DoS attack) and vulnerability indicators of ICANN. Algorithm-5 does a recursion validation check, this parameter should be checked for all the authoritative name servers of a domain. The recursive query is sent using the dig command to each authoritative name server and the response is recorded in variable MSG, which reveals whether recursion is enabled or not on authoritative name servers. If recursion is disabled in all the authoritative name servers of a given domain then this results in a recursive query check pass for that domain. The recursive query check should be performed against all authoritative DNS servers of the domain in concern.

Algorithm 5: Recursive Query Check

```
Input  : Name of the domain - DOMAIN
Output: 0 (Test Failed) or 1 (Test Passed)
1  Initialize NSCOUNT, RSCOUNT to 0
2  SERVERLIST= List of all DNS servers of DOMAIN
3  for I=1 to number of authoritative DNS Server of DOMAIN do
4      NSCOUNT = NSCOUNT + 1
5      MSG = send recursive query to next DNS server in the SERVERLIST[I]
6      if MSG contains "WARNING: recursion requested but not available" then
7          RSCOUNT = RSCOUNT + 1
8      end
9  end
10 if NSCOUNT=RSCOUNT then
11     return 1
12 end
13 else
14     return 0
15 end
```

3.2.1.6 Reverse Lookup Validation

A forward lookup maps an IP address to a domain name, whereas a reverse lookup maps a domain name to an IP address. A new TLD called "in-addr.arpa" was created to manage reverse lookups, and subdomains inside the "in-addr. arpa" domain are created by reversing the octets that make up an IP address. The reverse lookup domain for the 223.31.121.0/28 network, for example, is “121.31.223.in-addr.arpa.”

Reverse lookups are extremely important for troubleshooting and security; certain mail

Algorithm 6: Reverse Lookup Check

```
Input  : Name of the domain - DOMAIN
Output: 0 (Test Failed) or 1 (Test Passed)
Initialize NSCOUNT, RVCOUNT to 0
SERVERLIST= List of all DNS servers of DOMAIN
for I=1 to number of authoritative DNS Server of DOMAIN do
    NSCOUNT = NSCOUNT + 1
    NAME_Value = name of DNS Server in SERVERLIST[I]
    IP_Value = lookup for A record at DNS server in SERVERLIST[I]
    HOSTNAME = lookup for PTR record for IP_VALUE
    if HOSTNAME = NAME_Value then
        RVCOUNT = RVCOUNT + 1
    end
end
if NSCOUNT=RVCOUNT then
    return 1
end
else
    return 0
end
```

servers will refuse to exchange Internet messages with other servers if reverse lookups are not maintained. A reverse DNS query can be used to determine whether malicious actors are utilizing the authoritative DNS server's IP address. Spammers transmit spam to addresses that do not have a DNS record. A reverse search of their IP address to valid domain names should be possible on all healthy authoritative name servers. Malicious individuals routinely register domains without first doing a reverse lookup, and on the other hand, genuine companies should use the DNS server's reverse lookup as a best practice. In order to analyze the health of any DNS server, we propose that the availability of this capability be checked. One of the core ideas of DNS is that DNS resolution must be precise. For example, the domain "www.nitk.ac.in" resolves to the IP address 14.139.155.216, and 14.139.155.216 should also resolve to "www.nitk.ac.in"; this is the principle of coherence. As a result, the ICANN's coherency indicator and the reverse lookup check parameter are linked. Algorithm-6 does reverse lookup validation check; NSCOUNT and RVCOUNT variables contain the total number of authoritative name servers of a domain and the number of authoritative name servers having reverse lookup check passed. All the authoritative name servers of a domain will be validated for a reverse lookup check. NAME_Value gets the name of the authoritative name server and IP_Value gets the IP address of that DNS server. The host command is executed to retrieve the domain name of the IP address (IP_Value) and the result is stored in HOSTNAME. Now HOSTNAME and NAME_Value are compared to validate the reverse lookup check. If the NSCOUNT equals RVCOUNT it proves all authoritative name servers of that domain are validated for reverse lookup and the test result returns 1.

3.2.2 DNS Latency Comparison with ICMP Latency

DNS, which converts a domain name to an IP address, is critical to the quality of almost all Internet applications and services. As a result, DNS query delay in network latency is a critical factor impacting end-user Internet quality [70]. Because DNS is an often overlooked source of network delay, DNS lookups can drastically slow down a user's browsing experience. The query Round Trip Time (RTT) is the time it takes a DNS query to resolve a domain name to an IP address. DNS latency is a crucial performance metric that determines how successful a DNS system is. When an authoritative server gets overwhelmed, DNS resolution queries and responses must be delayed, and packets may be lost and resent. This is due to DNS server under-provisioning, which can result in a DoS assault. We aim to compare DNS query and ICMP echo request delay to see if DNS is under-provisioned or network congestion between the DNS client and authoritative DNS server.

The IP header includes an 8-bit TTL value that specifies the maximum number of layer-3 hops made on the route to their destinations. Before being transmitted further, a packet's TTL value is reduced by one as it passes through intermediated layer-3 devices. The TTL value determines the path between two network devices in the end. However, owing to the nature of the Internet, the two end devices can communicate via alternate channels in the future. As a result, the IP TTL values must be equal to compare DNS query and ICMP echo request delay, suggesting that the communications follow the same path.

Algorithm 7: DNS Query RTT vs. ICMP Echo RTT Check

```
Input  : DNS-Server-IP DNS-SERVER
Output: 0 (Test Failed) or 1 (Test Passed) or 2 ( Filtered by Firewall or
        different route)
Initialize TIMEOUT_DNS, TIMEOUT_ICMP by 0
DNS_TTL, DNS_RTT=Send a DNS query to DNS-SERVER and record the
  TTL values of the reply and DNS query Round Trip Time.
if TIMEOUT received in DNS query then
    TIMEOUT_DNS =1
end
ICMP_TTL, ICMP_RTT=Send a ICMP echo request to DNS-SERVER and
  record the TTL values of the reply and Round Trip Time of ICMP echo
  request.
if TIMEOUT received ICMP echo request then
    TIMEOUT_ICMP=1
end
if TIMEOUT_DNS=0 and TIMEOUT_ICMP=0 then
    if DNS_TTL = ICMP_TTL then
        if DNS_RTT >ICMP_RTT then
            return 0
        end
        else
            return 1
        end
    end
    else
        return 2
    end
end
else
    return 2
end
```

When a DNS query is transmitted to an authoritative DNS server, the DNS query RTT is recorded together with the IP header TTL value. Subsequently, an ICMP echo request is made to the authoritative DNS server, and the RTT and IP TTL values are recorded. When both TTL values are the same, both messages have taken the same route. Suppose the RTT for a DNS query is longer than the RTT for an ICMP echo request. In that case, this is the result of DNS under-provisioning and bad DNS health, but if the RTT for an ICMP echo request is longer than the RTT for a DNS query, the underlying network is congested, and DNS health is good. This newly discovered health measure matches ICANN's speed and stability indications.

The difference between DNS query RTT and ICMP echo RTT can indicate if the DNS server responds slowly (due to a DoS attack) or if the underlying network between the vantage point and the DNS server is congested. As a result, comparing DNS query latency and ICMP echo request latency is essential in determining DNS server health. Because the DNS design employs anycast for ROOT and TLD name servers, even if the authoritative server is anycast enabled, we always verify the IP TTL values of DNS and ICMP responses. If they match, we know the DNS query and ICMP echo request have the same destination.

3.2.3 DNSSEC Validation

When a DNS client makes a query to an authoritative DNS server, the client has no way of confirming the authenticity and integrity of the DNS reply because security was not a consideration when DNS was established. Since DNS utilizes UDP as its transport layer, DNS clients cannot detect a false or spoofed answer to one of their DNS requests. An attacker can easily spoof the authoritative server that a client requested by faking a response that appears to originate from that server. In other words, an attacker can redirect a user to a potentially dangerous website without the user's awareness. An IETF research team created the DNS Security Extensions (DNSSEC) in the 1990s to defend against this problem. DNSSEC is a DNS extension that adds security to DNS by preventing difficulties with integrity and authentication. As a result, DNSSEC protects against MITM attacks, DNS cache poisoning, and response manipulation by ensuring that the correct servers transmit the right responses. DNSSEC improves DNS response authentication by using digital signatures based on public-key cryptography. Authoritative DNS servers cryptographically sign DNS responses to ensure validity & integrity, and recursive resolver validates it. Because a domain is vulnerable to a DNS response spoofing attack if DNSSEC is not enabled, this value is critical for assessing the health of the authoritative DNS server. DNSSEC validation fulfills ICANN's integrity indicator because it preserves the integrity of DNS replies, verified denial of existence, and origin authentication. For a given domain name, Algorithm-8 does a DNSSEC validation check by downloading the latest root domains DNSSEC keys, which will be used to validate DNSSEC for a domain. The DNS query is sent to the local DNS server for asking DNSSEC records for the domain in question and the result is recorded into the MSG variable. If the MSG variable contains "SUCCESS" then DNSSEC is enabled for that domain. If MSG contains "FAILED" then DNSSEC is not enabled, else DNSSEC is not validated for that domain due to broken trust of chain in the DNS hierarchy.

Algorithm 8: DNSSEC Validation Check

```
Input  : Name of the domain - DOMAIN
Output: 0 (Test Failed) or 1 (Test Passed)
keys=download root DNSSEC Keys
MSG=send a DNS query asking DNSSEC check for DOMAIN to local DNS
  server
if MSG contains "SUCCESS" then
    DNSSEC=1
end
else if MSG contains "FAILED" then
    DNSSEC=0
end
else
    DNSSEC=2
end
if DNSSEC = 1 then
    return 1
end
else
    return 0
end
```

3.3 Authoritative Name Server Health Measurement and Key Finding

Because each discovered parameter is crucial for assessing health and poses a distinct amount of threat to health, the criticality of each discovered parameter is translated to a weightage as indicated in the Figure 3.3a.

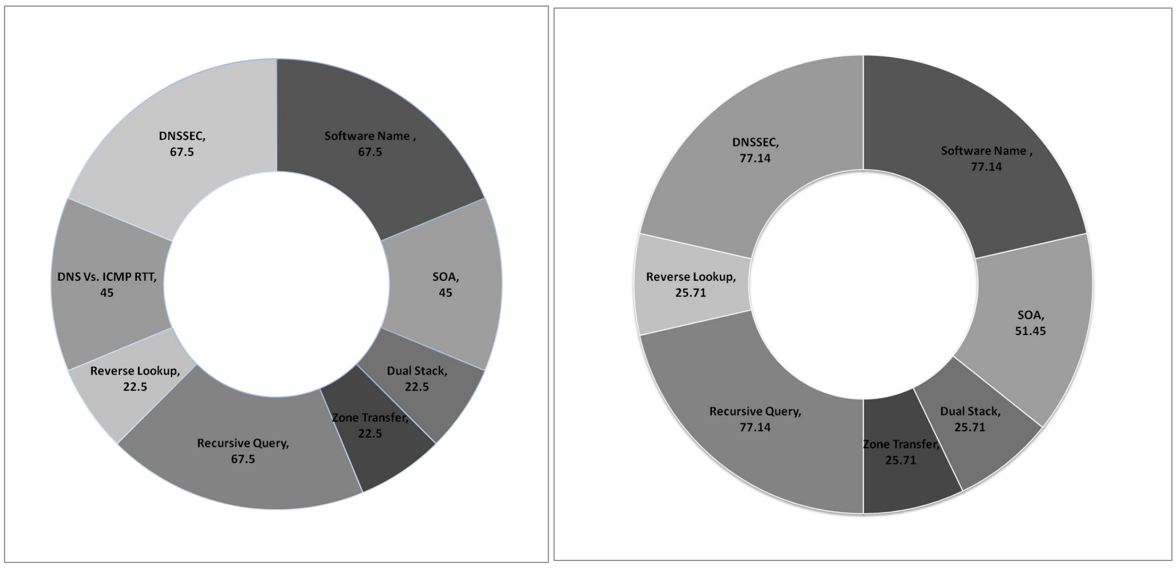
As many organizations block ICMP echo queries for security reasons, therefore we skip the DNS latency comparison with ICMP latency parameter check when evaluating DNS health in these instances. The revised weightage for the remaining parameters is calculated as shown in the Figure 3.3b. When we represent each parameter in a 360-degree form, we get a value of 45. If an ICMP timeout occurs while comparing the DNS query to ICMP echo RTT, and the value of each parameter will be 51.45. Table 3.1 shows the weightage computation for each proposed parameter.

All discovered parameters are assessed using algorithms 1 to 8 to determine the percentage of a domain's health, divided into four categories as shown in Table 3.2. We measured the health of authoritative DNS servers of the top 500 domains listed by Moz's ranking [83] for five consecutive days and recorded the result as shown in the Figure 3.4.

The health of the authoritative name server for the top 482 domains (remaining 18 are

Table 3.1 DNS Health Parameters Weightage Calculation

Proposed Health Parameters	Health Indicators from ICANN and MeNSa	Criticality	Weight	Value	Value When ICMP Timeout
Software Name Check	Availability and Vulnerability	High	1.5	67.5	77.14
SOA Check	Availability and Resiliency	Medium	1.0	45	51.45
Dual-Stack Check	Availability and Stability	Low	0.5	22.5	25.71
Zone Transfer Check	Vulnerability	Low	0.5	22.5	25.71
Recursive Query Check	Resiliency and Vulnerability	High	1.5	67.5	77.14
Reverse Lookup Check	Coherency	Low	0.5	22.5	25.71
DNS Query Vs. ICMP RTT Check	Speed and Stability	Medium	1.0	45	NA
DNSSEC Validation	Integrity	High	1.5	67.5	77.14
Total			8	360	360



(a) When ICMP Reply Received (b) When ICMP Timeout Received

Figure 3.3 Weightage of Identified Parameters as per Criticality

the name of the websites) is summarized in Table 3.3.

The health for the domain "example.com" is 100% on day-5, and it has been a healthy domain for all five days of testing, whereas the health percentage for the domain "time.com" has been 93% for all five days of testing. The domain "gnu.org" is consistently regarded as an unhealthy domain, with a healthy percentage of roughly 19.

The DNS software fingerprinting is revealed by almost 50% of the domain's authoritative name servers on all five days, as seen in Figure 3.5.

61% of the domains failed the SOA check, leaving them open to zone trashing and zone drift, which cause availability issues on them. A total of 41% of domains do not support dual-stack and are hosted only on the IPv4 network. An average of 163 do-

Table 3.2 DNS Health Categories

S/No.	Health Percentage	Health category
1.	75-100	Healthy
2.	50-74	Moderately Healthy
3.	25-49	Poor Health
4.	Below 25	Unhealthy

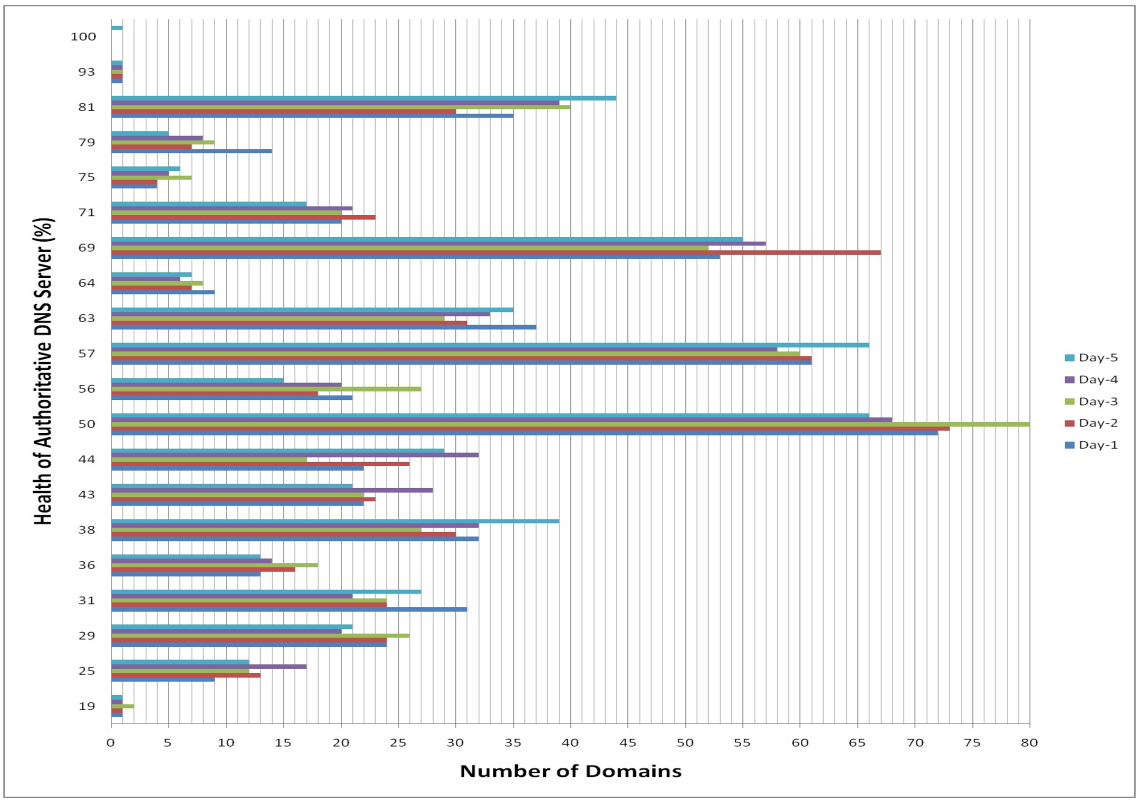


Figure 3.4 A Report on the Health of the Authoritative Name Servers for the Top 500 Domain Names

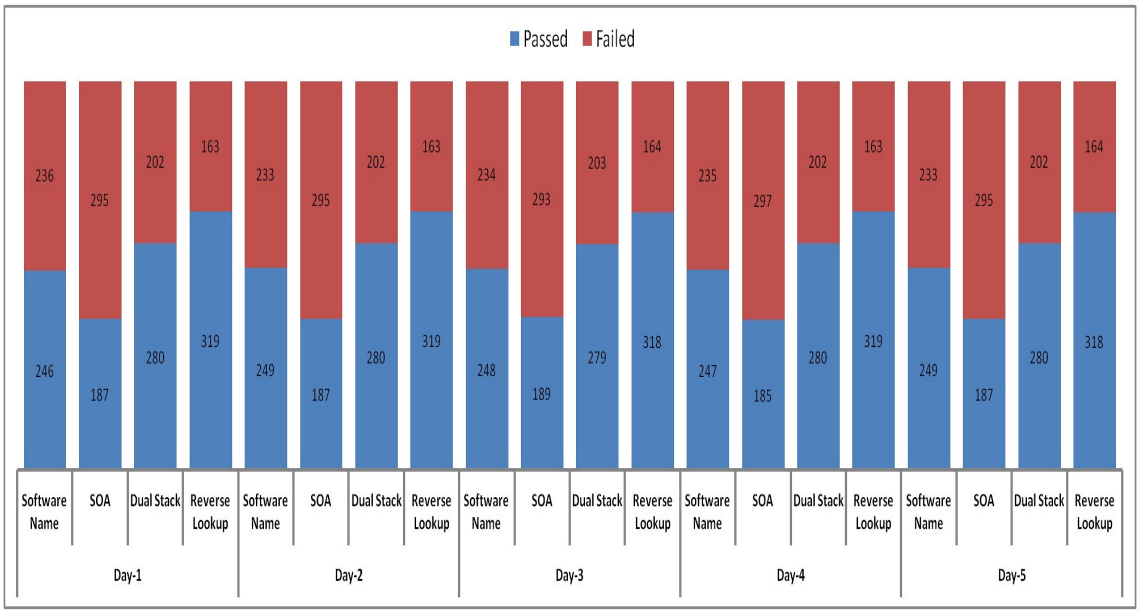


Figure 3.5 Software Name, SOA, Dual Stack and Reverse Lookup check for the Five Days of Health Assessment

Table 3.3 DNS Health Results

Domains Health/Days	Day-1	Day-2	Day-3	Day-4	Day-5
Healthy	55	44	58	53	57
Moderately Healthy	273	280	276	263	261
Poor Health	153	156	146	164	162
Unhealthy	1	2	2	2	2

mains do not adhere to the coherency principle i.e. not configured for reverse lookup.

The Figure 3.6 shows that zone transfers are permitted on three domain's authoritative name servers on all five days. These domains have vulnerabilities and are prone to leaking potentially sensitive information, and an attacker can gain knowledge of the infrastructure, allowing him to plan more complex network-based attacks. For all five days, almost all domain authoritative name servers prevent recursive requests, and only three authoritative name servers (adobe.com, mail.ru, and samsung.com) serve as recursive servers, making them vulnerable to DDoS. Even though DNSSEC has been around for 15 years, our findings demonstrate that it is still not widely used or obeyed by authoritative name servers; our result shows that only 33 domains out of 482 are DNSSEC enabled.

The DNS latency comparison with the ICMP latency parameter check is an important health indicator. This comparison may reveal that DNS servers can be under DoS attack (if latency difference is too high), proving that the health results can be computed dynamically using our methodology. As shown in Figure 3.7, the results vary from time to time, and therefore a healthy server today may not be healthy tomorrow, indicating a probable attack. This could be used to alert the respective administrator to take measures to prevent such attack scenarios. Therefore the extended parameters (specifically DNS query vs. ICMP echo RTT) are better in evaluating the authoritative name servers effectively and accurately.

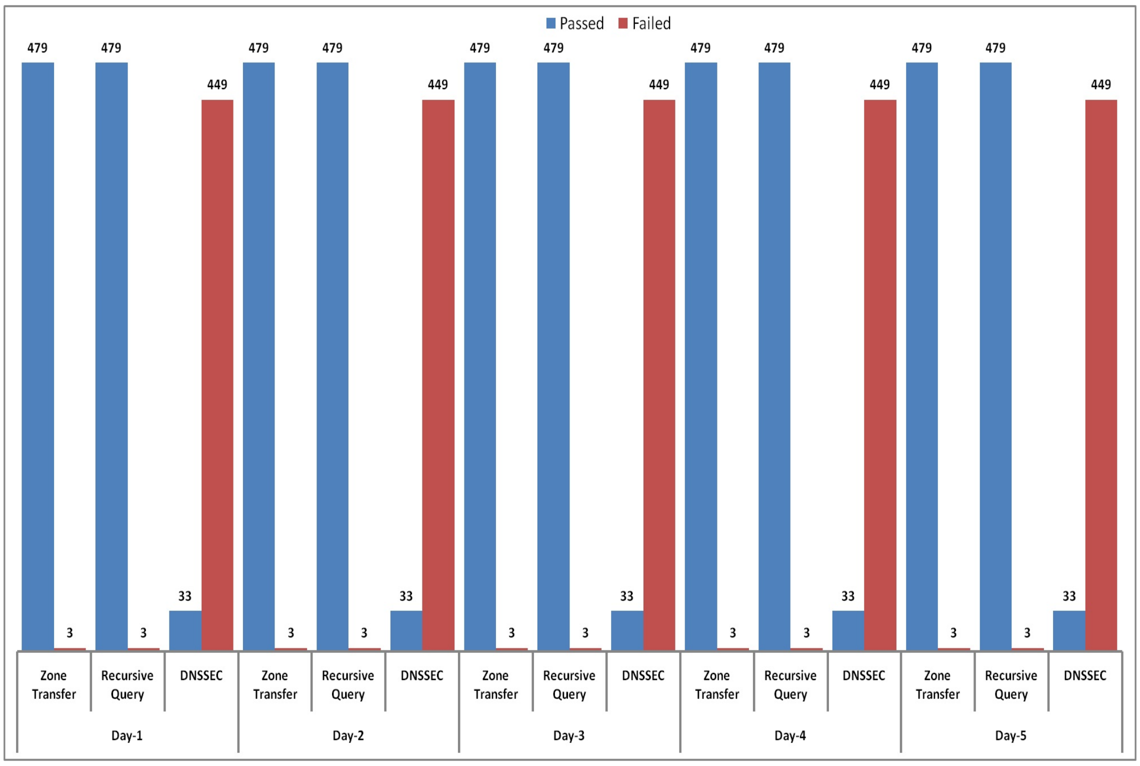


Figure 3.6 Zone Transfer, Recursive Query, and DNSSEC check for the Five Days of Health Assessment

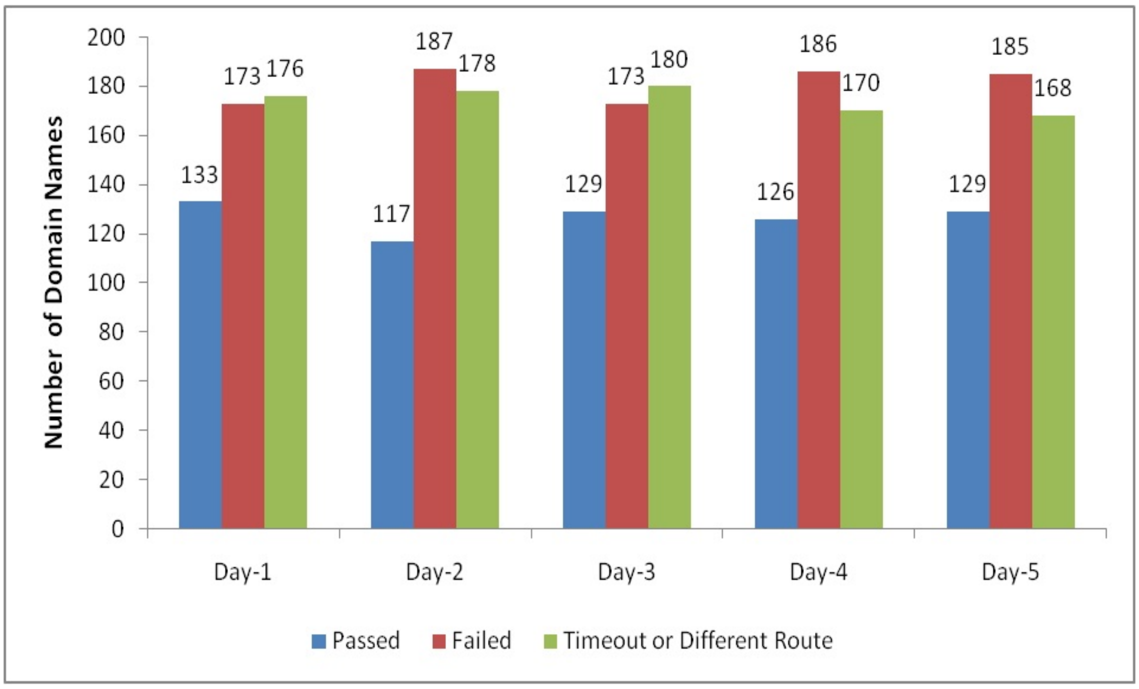


Figure 3.7 Results of DNS Query vs. ICMP Echo RTT Check

3.4 Summary

We identified 8 novel parameters (Software Obfuscation, SOA Validation, Dual-Stack Support, Zone Transfer Status, Non-Recursive Querying, Reverse Lookup Validation, Latency Comparison of DNS Query and ICMP Echo Request, and DNSSEC Validation) and proposed a series of algorithms for measuring the health of authoritative DNS servers. The results obtained on a list of the authoritative name servers serving the top 500 domains for five days have been tabulated. By measuring the health of the top 500 domain's authoritative name servers, we highlight the critical problem areas for those name servers serving the domains that need to be resolved by addressing the appropriate health parameters. This passive approach can be scaled easily across the DNS hierarchy and could be used to determine the health of the global or regional or even a segment of the DNS system at any given instance of time. The experiment can be repeated periodically to identify divergent behavior that may lead to malicious attacks on the Internet infrastructure.

Chapter 4

DNS Intrusion Detection (DID): IDS Signatures for DNS Tunneling, Amplification, and DoS Attacks

4.1 Introduction

The Internet is the global connectivity of multiple networks managed by the various stakeholders like government and private organizations, business, academia, ISP etc. Initially designed to connect government research centers (ARPA network), the Internet has grown exponentially since 1994 to serve billions of users and electronic gadgets connected to every corner of the globe. As there are more than 1.8 billion websites on the Internet and among them around 400 million live at any point in time, the DNSEXT of the IETF has designated DNS as the "Critical Infrastructure" due to the ever-expanding Internet and its phenomenal growth of websites. As a result, DNS is the most targeted protocol among all Internet applications in today's Internet era. DNS abuses are classified into two categories: attacks on DNS infrastructure and attacks exploiting DNS infrastructure. Many abuses in the first category, i.e., attacks on DNS infrastructure, have already been addressed by DNSSEC and TSIG protocols, except for DDoS attacks. However, many challenges like DNS amplification and tunneling are still open for researchers in the second type of attack, i.e., attacks exploiting DNS infrastructure. DNS tunneling can use DNS infrastructure as a formidable weapon to attack any organization for data exfiltration or command and control (C2). Oskar Pearson suggested DNS tunneling on the Bugtraq mailing group in 1998, and in the early 2000s, a tool called NSTX was created as a proof of concept for DNS tunneling [84]. Since then, several tools have been built, and malware such as Morto and Feederbot use it to convert command and control channels [71]. Most organizations do not monitor DNS traffic for suspicious activities. DNS is usually permitted to freely move through perimeter security devices like firewalls, making DNS an attractive vector for attackers that use DNS tunneling to convey commands and exfiltrate data discretely. A wide range of tunneling technologies via DNS is developed, and they are explored in this thesis work.

On the other hand, DNS amplification uses the benefits of open recursive resolvers to overwhelm the network of any victim on the Internet. Since the inception of the Internet, this issue is still open for researchers to work on. The Internet has previously been abused several times using DNS amplification attacks, including the Dyn attack in 2016 and the Spamhaus attack in 2013. Many security applications, appliances, protocols, and DNS security add-ons have emerged over the recent year, and the DNS protocol has been updated several times. When it comes to DNS security, traditional network security technologies such as firewalls and intrusion detection & prevention systems have shown to be inadequate. They provide insufficient coverage of the DNS threat environment, leading to an unacceptably high number of false positives and, in certain cases, system failures. Finally, they cannot identify data exfiltration efforts or DNS tunneling, reflective amplification in a timely way, which is critical in today's highly regulated contexts [72].

DNSSEC is a set of IETF protocols for ensuring the security of DNS information. It is simply a set of DNS extensions that offer authentication of response/query, response integrity, and authentication denial of existence. DNSSEC protects the malicious user from tampering with DNS record answers, which might lead to visitors being sent to their website, thus protecting against MITM attacks. The additional security provided by DNSSEC comes at a cost since attackers can use the higher domain sizes to launch DNS amplification attacks [73].

Transaction Signature (TSIG) is developed to protect the transaction of zone transfer between master and slave DNS servers. The use of symmetric keys and cryptographic hash functions in TSIG guarantees safe communication between masters and slave servers. TSIG assures that data received in zone transfers is valid and has not been tampered with in transit and the DNS response's legitimacy. TSIG is a technique for preventing IP spoofing during resource record (RR) updates between master and slave servers, and it is used to authenticate DNS database updates. There has been much research done and many security products and protocols established for safeguarding DNS infrastructure. However, none of them can stop all DNS-related attacks, and there are still many security vulnerabilities with DNS. We propose a snort-based approach to counter all probable DNS-related threats in this research work.

4.2 Proposed Methodologies

Multiple technologies exist for securing DNS infrastructure against various Internet threats, with TSIG, DNSSEC, and IDS being the most popular and extensively implemented. TSIG is only used to secure zone transfers between master and slave servers and ensure that cryptographically authenticated slaves can download critical zone files from the master server. In contrast, DNSSEC ensures the integrity and authenticity of the DNS transaction between authoritative DNS servers and recursive resolvers. Both TSIG and DNSSEC do not completely secure the DNS protocol from DNS abuses such as DNS tunneling, amplification, and denial-of-service attacks. The purpose of IDS approaches, on the other hand, is to examine acceptable and unacceptable behavior of DNS traffic in order to identify threats to the DNS protocol. Even though IDS solutions include a diverse signature set for practically detecting all Internet threats, there are only a few rules for identifying abuses of DNS infrastructure, and those rules are generic and not tailored to a specific attack weapon or application. This research work fills the gaps by developing new IDS signatures for practically all tools used for DNS tunneling, amplification, and DoS attacks. These novel signatures are combined with SNORT's current DNS-based signatures to protect any node in the DNS infrastructure hierarchy, including Root, TLDs, authoritative servers, recursive servers, and even stub resolvers. The resultant IDS is called DNS Intrusion Detection - DID.
An intrusion in a network is a malicious activity that violates any security principles, such as privacy, integrity, availability, and authentication. Network traffic must be monitored for particular network segments or devices to defend against network intrusions.

TCP/IP network, transport, and application layer protocols must be examined to detect malicious activities. This job is done by IDS, which sends out an alert whenever it detects malicious activity on the network. As the name implies, IDS can only identify ongoing security abuses in the network by raising a specific alarm to notify the right persons. Many free and open-source IDS systems are available, including SNORT, SURICATA, BRO, and Security Onion. This research work proposes an approach called DID for safeguarding DNS infrastructure utilizing SNORT IDS and a novel IDS signature applied to SNORT to defend almost all kinds of DNS attacks, including DNS tunneling, DNS amplification, and DNS DoS/DDoS attacks. We have considered SNORT in this work as it is among the most popular and familiar IDS. SNORT Talos is a well-known signature-based network IDS with many rules for identifying malicious programs and unusual network activities [74].

4.2.1 DNS Tunneling

The DNS protocol is utilized by all devices connected to the Internet. Most organizations do not monitor DNS traffic, making it a convenient target by attackers for Internet abuse. DNS tunneling is a technique in which an adversary utilizes DNS query and response to tunnel any communication, such as HTTP, FTP, SSH, and even the complete internet protocol (IP) traffic of TCP/IP. Malicious users primarily utilize the DNS tunnel for data exfiltration and command and control (C2). DNS tunneling was first introduced in 1998, and since then, various tools have been created to enable DNS tunneling, as indicated in Table 4.1.

Suppose all outgoing DNS (UDP port 53) traffic is allowed through the organization's edge firewall. In that case, a DNS tunnel can be established between a compromised client and the attacker's controlled DNS server, allowing data exfiltration or C2 even if all other traffic is blocked at the firewall except DNS. When a tunnel is successfully established between a compromised client and an attacker's DNS tunnel server, certain tunneling tools create TUN or TAP network interfaces on both endpoints of the tunnel. This interface can also carry any protocols traffic between two endpoints, and even entire IP traffic can be tunneled through a DNS tunnel.

4.2.1.1 The Component of the DNS Tunnel

The DNS tunnel consists of two primary components: a compromised DNS client and the attacker's authoritative DNS server. The attacker must register an Internet domain or subdomain. The domain's authoritative DNS must refer to the attacker's DNS server; this is where the server portion tunneling software is configured and running. The attacker needs to install the client portion of tunneling software on the victim's system. Upon successful installation, the compromised client sends a DNS query for a subdomain representing an encoded communication. A local or ISP DNS resolver at the client-side finally routes the query to the attacker's DNS server. The compromised client then receives a malicious DNS response from the controlled DNS server, which contains commands to the clients for data exfiltration or C2, and passes unnoticed via perimeter devices like firewalls. Through the DNS tunnel, the attacker may keep doing this over time and being undetected.

As shown in Figure 4.1, to construct a tunnel, the attacker utilizes data encoded in the DNS payload. DNS tunneling transports IPv4 network packets over DNS queries

Table 4.1 List of DNS Tunneling Tools

Name	Programming Language	Resource Record Used	Encoding	Platform	Encryption
Iodine	C	NULL	Base32/64	Linux, Unix, MacOS X and Windows	No
DNSCAT2	C and Ruby	CNAME, MX , TXT	HEX and NetBIOS	Linux, Unix, MacOS X and Windows	Yes
DNS2TCP	C	TXT, KEY	BASE64	Linux	No
ThunderDNS	Python, Bash, powershell	TXT	Base64	Windows and Unix	No
OzmanDNS	Perl	TXT	Base32	Linux	No

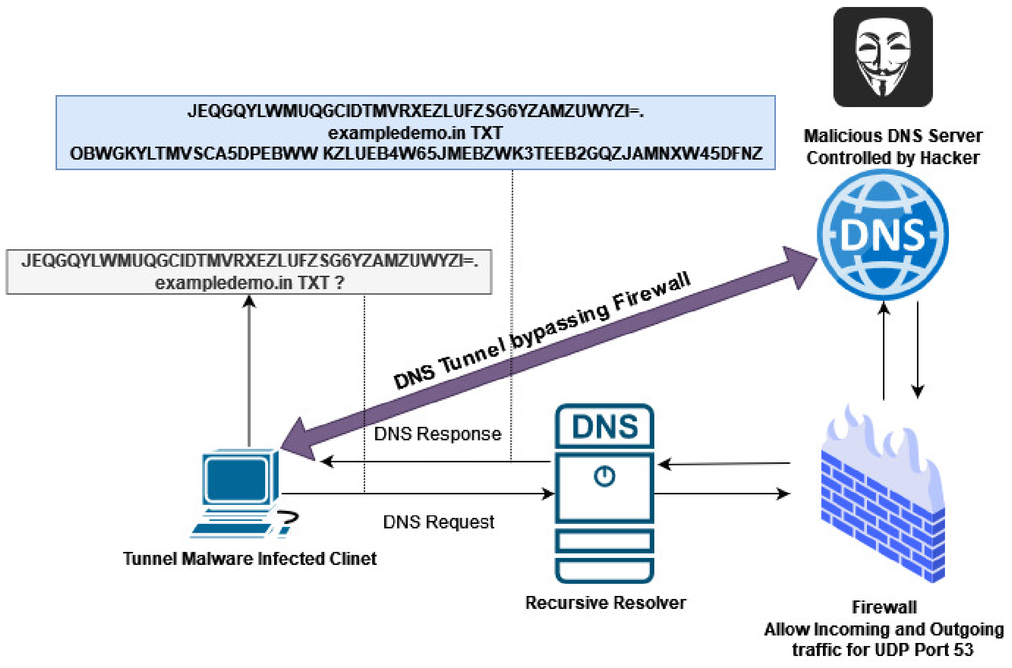

JEQGQYLWMUQGCIDTMVRXEZLUFZSG6YZAMZUWYZI=.
exampledemo.in TXT
OBWGKYLTMVSCA5DPEBWW KZLUEB4W65JMEBZWK3TEEB2GQZJAMNXW45DFNZ
Malicious DNS Server Controlled by Hacker
JEQGQYLWMUQGCIDTMVRXEZLUFZSG6YZAMZUWYZI=.
exampledemo.in TXT ?
DNS Tunnel bypassing Firewall
DNS
DNS Response
DNS
DNS Request
Tunnel Malware Infected Clinet
Recursive Resolver
Firewall
Allow Incoming and Outgoing traffic for UDP Port 53

Figure 4.1 DNS Tunneling Attack

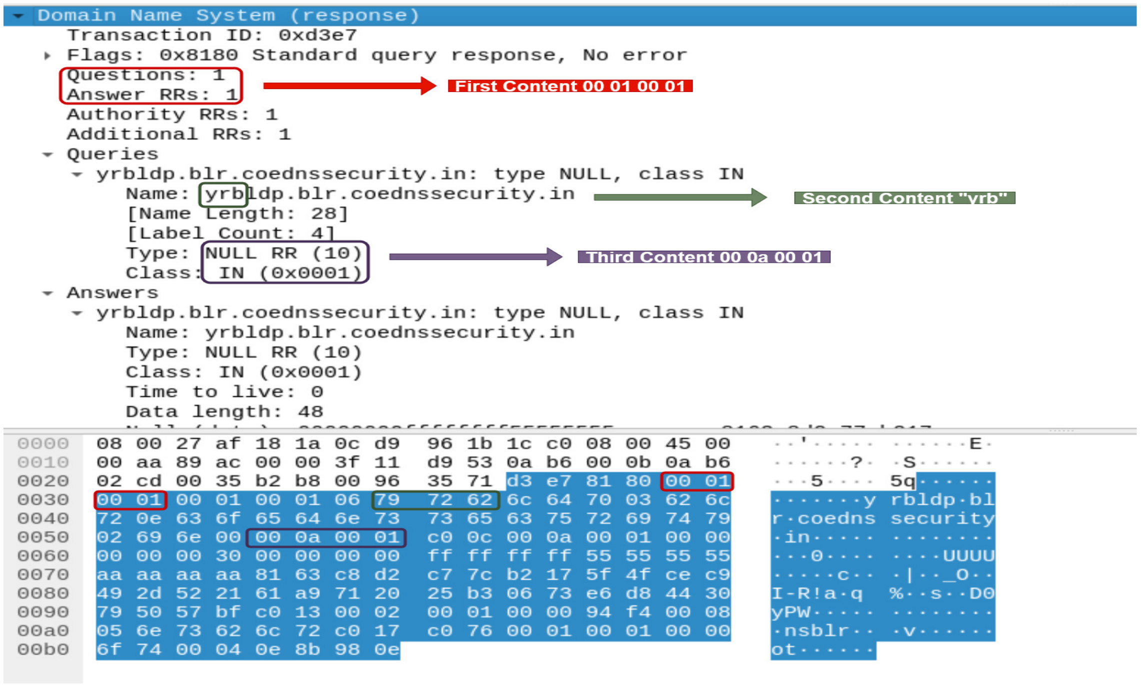
and responses by using the hostname to convey data using a DNS query and a record type, e.g., MX, A, CNAME, TXT, and NULL for transferring the answer, implying that DNS queries will be in a format similar to <encoded data>.exampledns.com and same is applied to DNS responses. Base32, Base64, Binary, HEX, and NETBIOS are some of the encoding formats that may encode data in the DNS queries and responses. Assume that the hacker has a compromised client and owns the domain "exampledemo.in.". To transport the data, hackers use base32 encoding method, which obscure and compress the data while constantly slicing it into arbitrary sizes. If the message "I have a secret.doc file" has to be delivered to the hacker's DNS tunnel server, the compromised client issues the following DNS query: "JEQGQYLWMUQGCIDTMVRXE-ZLUFZSG6YZAMZUWYZI=.exampledemo.in TXT ?". Once the query is being received by hackers DNS tunnel server, If the hacker wishes to send the message "pleased to meet you, send the content of the secret.doc file," it sends the following response " JE-QGQYLWMUQGCIDTMVRXEZLUFZSG6YZAMZUWYZI=.exampledemo.in TXT OBWGKYLTMVSCA5DPEBWW KZLUEB4W65JMEBZWK3TEEB2GQZJAMNXW45DFNZ". After getting the answer, the compromised client will use the same technique to exfiltrate the content of a secret.doc file. This is how DNS tunneling works and how it is used for data exfiltration. Many tools are available on the Internet to demonstrate the notion of DNS tunneling, each with its own set of features [75], with the most commonly used being: Iodine, DNSCAT2, DNS2TCP, ThunderDNS, and OzmanDNS. To defend DNS tunneling, we recommend that all of these tools have an IDS signature, which we propose in this research work.

4.2.1.2 IDS Signature for Iodine

Bjorn Andersson and Erik Ekman [85] created the Iodine application in C programming language, using a client-server architecture to tunnel IPv4 network traffic using DNS queries and responses. The tool uses NULL record type and supports fragmentation, compression, and encoding. The whole IP traffic is tunneled by establishing a TUN interface on the client and server components assigned to an IPv4 address. In this way, any application layer protocol can be wrapped by an IP header is exchanged between client and server. It also uses the EDNS protocol to transport network packets larger than 512 bytes over UDP, resulting in improved speed. The tool can operate on various UNIX-like platforms & Windows, and tunnels may be established between any two hosts regardless of the operating system.
In the iodine tunnel, the NULL resource record type is frequently utilized. The NULL RR type is no longer in use and is now solely used for testing. If NULL is encountered in a DNS query/response, DNS tunnel activity is suspected. The Figure 4.2 shows a typical network packet captured for Iodine tunnel traffic.
Based on the characteristic and traffic captured of iodine application we have developed following IDS signature to recognise iodine tunnel:

IDS Signature for Iodine

```
alert udp any 53 -> any any (msg:" iodine DNS Tunnelling Request"; content:"|00 01 00 01|", offset 4, depth 4; content:"yrb", distance 4, within 4; content:"|00 0a 00 01|", within 255; threshold: type threshold, track by_src, count 2, seconds 15; sid: 806254370; rev:1;)
```



```
▾ Domain Name System (response)
    Transaction ID: 0xd3e7
  ▸ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 1
    Additional RRs: 1
  ▾ Queries
    ▾ yrbldp.blr.coednssecurity.in: type NULL, class IN
        Name: yrbldp.blr.coednssecurity.in
        [Name Length: 28]
        [Label Count: 4]
        Type: NULL RR (10)
        Class: IN (0x0001)
  ▾ Answers
    ▾ yrbldp.blr.coednssecurity.in: type NULL, class IN
        Name: yrbldp.blr.coednssecurity.in
        Type: NULL RR (10)
        Class: IN (0x0001)
        Time to live: 0
        Data length: 48

0000  08 00 27 af 18 1a 0c d9  96 1b 1c c0 08 00 45 00   ··'·····  ······E·
0010  00 aa 89 ac 00 00 3f 11  d9 53 0a b6 00 0b 0a b6   ······?·  ·S······
0020  02 cd 00 35 b2 b8 00 96  35 71 d3 e7 81 80 00 01   ···5····  5q······
0030  00 01 00 01 00 01 06 79  72 62 6c 64 70 03 62 6c   ·······y  rbldp·bl
0040  72 0e 63 6f 65 64 6e 73  73 65 63 75 72 69 74 79   r·coedns  security
0050  02 69 6e 00 00 0a 00 01  c0 0c 00 0a 00 01 00 00   ·in·····  ········
0060  00 00 00 30 00 00 00 00  ff ff ff ff 55 55 55 55   ···0····  ····UUUU
0070  aa aa aa aa 81 63 c8 d2  c7 7c b2 17 5f 4f ce c9   ·····c··  ·|··_O··
0080  49 2d 52 21 61 a9 71 20  25 b3 06 73 e6 d8 44 30   I-R!a·q   %··s··D0
0090  79 50 57 bf c0 13 00 02  00 01 00 00 94 f4 00 08   yPW·····  ········
00a0  05 6e 73 62 6c 72 c0 17  c0 76 00 01 00 01 00 00   ·nsblr··  ·v······
00b0  6f 74 00 04 0e 8b 98 0e                            ot······
```

Figure 4.2 A Network Packet Capture for Iodine Traffic

The above signature will detect iodine traffic if the DNS response packet contains "00 01 00 01" (both DNS questions and Answer Resource Records contains 1 each), and "yrb" (As in the initial client request, the domain is prefixed with "yrb" in the DNS query, the response also contains this prefix to the domain name) and "00 0a 00 01" (represents NULL type request and IN class). This pattern should match at least twice in fifteen seconds to generate an alert for iodine tunnel traffic.
The client tries to auto-detect DNS query type, and if it gets NOTIMP as a reply which means the server does not support this kind of request, then CNAME is used as default query type, and the domain is prefixed with "yrb." The default snort signature will not be able to detect this; thus, we identified the following signature as:

IDS Signature for Iodine

alert udp any 53 -> any any (msg:" iodine DNS Tunnelling Request"; content:"|00 01 00 01|", offset 4, depth 4; content:"yrb", distance 4, within 4; content:"|00 05 00 01|", within 255; threshold: type threshold, track by_src, count 2, seconds 15; sid: 806254370; rev:1;)

4.2.1.3 IDS signature for DNSCAT2

DNSCAT2 is acclaimed for its ability to construct a command and control tunnel using the DNS protocol, allowing an adversary to operate invisibly. The main advantage of DNSCAT2 over other tools is that tunnel traffic is encrypted, and also the attacker does not require an authoritative DNS server for a domain name. The tunnel may be precisely established among the compromised victim and tunnel server over UDP port 53, used for file downloads, uploads, and even remote access shells. DNSCAT2 has two parts: a server and a client. The server part is made in a ruby programming language and needs to run a DNS tunnel server over UDP port 53. It tries to establish a legitimate association when it receives DNS queries from the compromised clients.
On the other hand, the client is part of DNSCAT2 written in "C" and intended to operate indefinitely on a target machine. When the client component is run on a compromised server with the domain name specified, the client negotiates with the tunnel server by sending DNS queries to the local DNS server, which then forwards them to the DNS tunnel server, which the adversary ostensibly controls. If the client is run without specifying a domain name, it will attempt to connect directly to the tunnel server through port 53.
If DNSCAT2 traffic is sent over an unencrypted channel, it may be readily observed. Because all queries and answers are always prefixed with "dnscat," its traffic is detectable. This traffic may be identified using the following IDS rule:

IDS signature for DNSCAT2

alert udp any any -> any 53 (msg: “DNSCAT2-Tunneling Attempt”; content:"|00 01 00 00|",offset 4,depth 4; content: “dnscat”, distance 28, within 255; sid: 806254374; rev:1)

DNSCAT2 uses CNAME, MX, and TXT resource records in DNS query and responses and Hex or NetBIOS encoding techniques. However, because DNSCAT2 employs encryption in client-server connections, it will not be easy to detect. The DNS query/reply packets are always bigger than 100 bytes because each packet comprises an unencrypted 5-byte header and an unencrypted 2-byte nonce. Keep in mind that normal

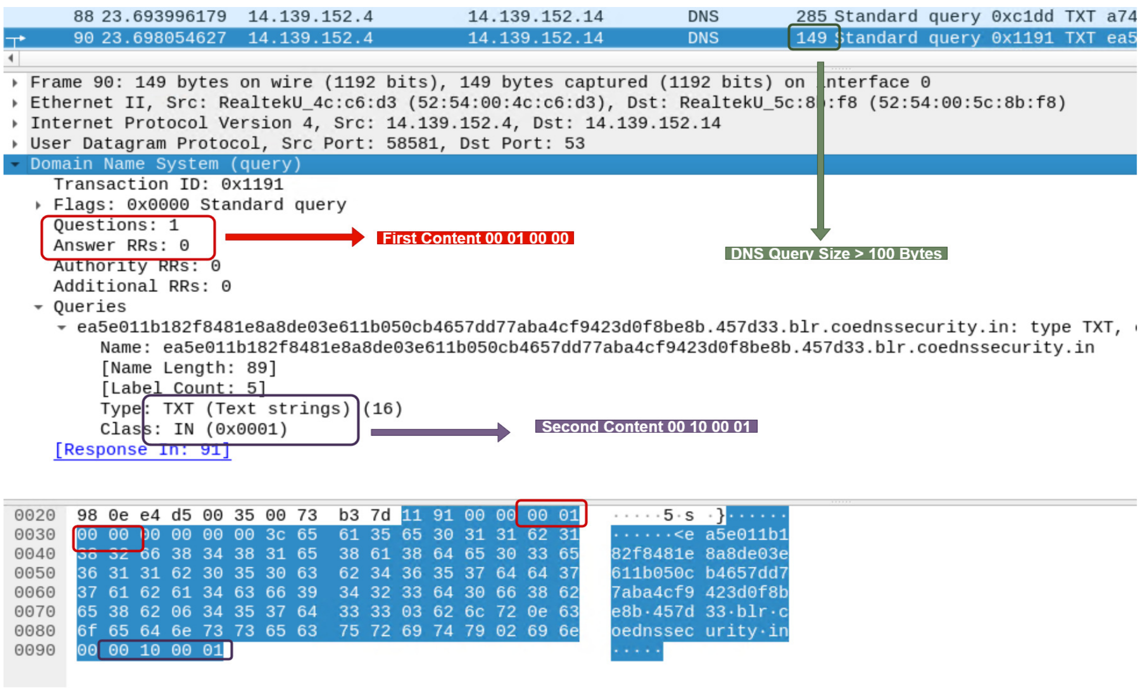
DNS packets are typically less than 100 bytes on average [86], whereas, for DNSCAT2 traffic, the average size of query or response packets is more than 100 bytes. Based on these characteristics of DNSCAT2 tunnel traffic, we created the following novel IDS signatures below for identifying DNSCAT2 tunnel CNAME traffic:

IDS signature for DNSCAT2

alert udp any any -> any 53 (msg: "Possible-DNSCAT2-Tunnel-CNAME-Traffic"; content:"|00 01 00 00|",offset 4,depth 4; content: "|00 05 00 01|",distance 6, within 255; dsize:>100;detection_filter: track by_src, count 2, seconds 10;sid:806254375;rev:1;)

The Figure 4.3 shows a typical network packet captured for DNSCAT2 tunnel traffic.

Suppose the DNS query packet is larger than 100 bytes and contains "00 01 00 00" (both DNS queries and Answer Resource Records contain one each) and "00 05 00 01" (represents CNAME type request and IN class). In that case, the signatures described


```
   88 23.693996179  14.139.152.4         14.139.152.14        DNS        285 Standard query 0xc1dd TXT a74
   90 23.698054627  14.139.152.4         14.139.152.14        DNS        149 Standard query 0x1191 TXT ea5

 Frame 90: 149 bytes on wire (1192 bits), 149 bytes captured (1192 bits) on interface 0
 Ethernet II, Src: RealtekU_4c:c6:d3 (52:54:00:4c:c6:d3), Dst: RealtekU_5c:8b:f8 (52:54:00:5c:8b:f8)
 Internet Protocol Version 4, Src: 14.139.152.4, Dst: 14.139.152.14
 User Datagram Protocol, Src Port: 58581, Dst Port: 53
 Domain Name System (query)
   Transaction ID: 0x1191
   Flags: 0x0000 Standard query
   Questions: 1
   Answer RRs: 0
   Authority RRs: 0
   Additional RRs: 0
   Queries
     ea5e011b182f8481e8a8de03e611b050cb4657dd77aba4cf9423d0f8be8b.457d33.blr.coednssecurity.in: type TXT,
       Name: ea5e011b182f8481e8a8de03e611b050cb4657dd77aba4cf9423d0f8be8b.457d33.blr.coednssecurity.in
       [Name Length: 89]
       [Label Count: 5]
       Type: TXT (Text strings) (16)
       Class: IN (0x0001)
   [Response In: 91]

0020  98 0e e4 d5 00 35 00 73  b3 7d 11 91 00 00 00 01   ·····5·s ·}······
0030  00 00 00 00 00 00 3c 65  61 35 65 30 31 31 62 31   ······<e a5e011b1
0040  38 32 66 38 34 38 31 65  38 61 38 64 65 30 33 65   82f8481e 8a8de03e
0050  36 31 31 62 30 35 30 63  62 34 36 35 37 64 64 37   611b050c b4657dd7
0060  37 61 62 61 34 63 66 39  34 32 33 64 30 66 38 62   7aba4cf9 423d0f8b
0070  65 38 62 06 34 35 37 64  33 33 03 62 6c 72 0e 63   e8b·457d 33·blr·c
0080  6f 65 64 6e 73 73 65 63  75 72 69 74 79 02 69 6e   oednssec urity·in
0090  00 00 10 00 01                                     ·····
```
First Content 00 01 00 00
DNS Query Size > 100 Bytes
Second Content 00 10 00 01

Figure 4.3 A Network Packet Capture for DNSCAT2 Traffic

above will identify iodine CNAME traffic; this pattern must match at least twice in ten seconds. Similarly following signatures detect DNScat2 tunnel MX and TXT traffic:

IDS Signatures for DNSCAT2

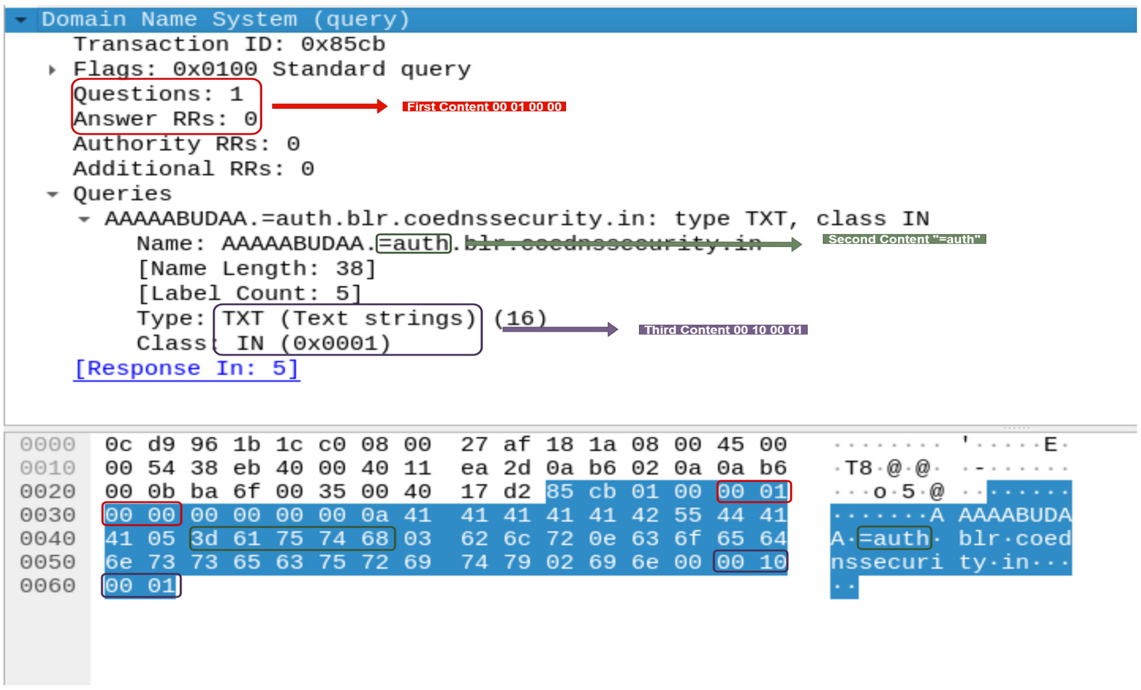
```
alert udp any any -> any 53 (msg: "Possible-DNSCAT2-Tunnel-MX-Traffic"; content:"|00 01 00 00|",offset 4,depth 4; content: "|00 0f 00 01|",distance 6, within 255; dsize:>100;detection_filter: track by_src, count 2, seconds 10;sid:806254376;rev:1;)
alert udp any any -> any 53 (msg: "Possible-DNSCAT2-Tunnel-TXT-Traffic"; content:"|00 01 00 00|",offset 4,depth 4; content: "|00 10 00 01|",distance 6, within 255; dsize:>100;detection_filter: track by_src, count 2, seconds 10;sid:806254377;rev:1;)
```

4.2.1.4 IDS Signature for DNS2TCP

The primary goal of the DNS2TCP application is to overcome the captive portal and gain free Internet access; hence it tunnels TCP traffic over the DNS protocol and does not require to have TUN/TAP interface. The advantage of this utility over others is that it can execute on client systems without requiring administrator privileges. The tool is written in C programming language and only accepts TXT or KEY requests, and the data sent is encoded in Base64. It utilizes the fixed subdomain "=auth" to initiate TCP handshakes between client and server and employs the session-tag, which uses the first four bytes of the subdomain to monitor and keep the same session. Based on the behavior of the DNS2TCP tunnel, we created the following signature to identify the tunnel traffic:

IDS signature for DNS2TCP

```
alert udp any any -> any 53 (msg: "DNS2TCP-TCP Handshake"; content:"|00 01 00 00|", offset 4, depth 4; content:"=auth", distance 4; content:"|00 10 00 01|", distance 2, within 255; sid:806254378; rev:1;)
```



```
▾ Domain Name System (query)
    Transaction ID: 0x85cb
  ▸ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▾ AAAAABUDAA.=auth.blr.coednssecurity.in: type TXT, class IN
        Name: AAAAABUDAA.=auth.blr.coednssecurity.in
        [Name Length: 38]
        [Label Count: 5]
        Type: TXT (Text strings) (16)
        Class: IN (0x0001)
    [Response In: 5]

0000  0c d9 96 1b 1c c0 08 00  27 af 18 1a 08 00 45 00   ········ '·····E·
0010  00 54 38 eb 40 00 40 11  ea 2d 0a b6 02 0a 0a b6   ·T8·@·@· ·-······
0020  00 0b ba 6f 00 35 00 40  17 d2 85 cb 01 00 00 01   ···o·5·@ ········
0030  00 00 00 00 00 00 0a 41  41 41 41 41 42 55 44 41   ·······A AAAABUDA
0040  41 05 3d 61 75 74 68 03  62 6c 72 0e 63 6f 65 64   A·=auth· blr·coed
0050  6e 73 73 65 63 75 72 69  74 79 02 69 6e 00 00 10   nssecuri ty·in···
0060  00 01                                              ··
```

Figure 4.4 A Network Packet Capture for DNS2TCP Traffic

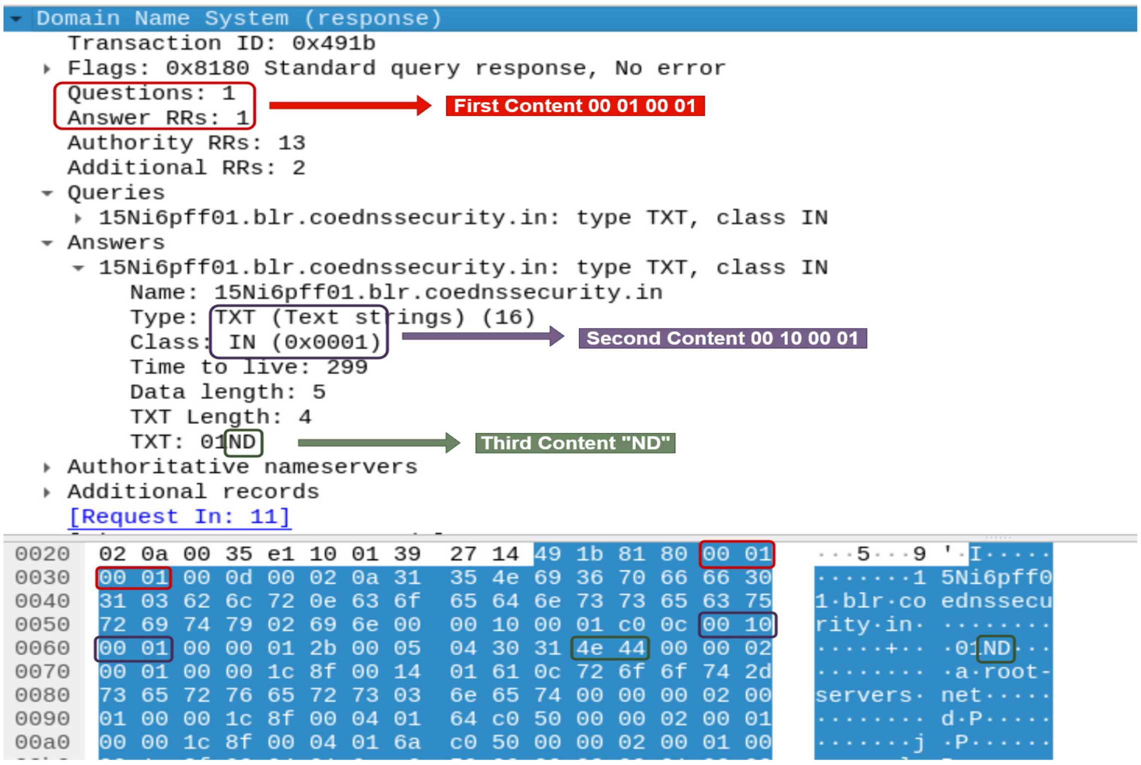
If a DNS query is transmitted over port 53 to any DNS server, the signature, as mentioned above, comprises three contents to detect DNS2TCP tunnel traffic. The first content "00 01 00 00" belongs to question resource records equals one and Answer Resource Records equals zero, the second content "=auth" correlates to the TCP handshake string between client and server, and the third content "00 10 00 01" corresponds to the IN class TXT record.We captured the network packet of DNS2TCP attack traffic as shown in Figure 4.4.

4.2.1.5 IDS Signature for ThunderDNS

ThunderDNS tunnels like DNS2TCP may be used to build a tunnel that routes TCP traffic over the DNS protocol. The tunnel server application is developed in Python and is even Docker-compatible. The client software is written in a Linux shell, a Windows Power Shell, and a PHP web application [87]. The majority of IDS do not have an inherent ThunderDNS tunnel signature.

For communication between the server and the client, it employs a basic protocol; the following are the features of client-server communication:

1. The client requests a TXT record in the following format from the server to register



```
▾ Domain Name System (response)
    Transaction ID: 0x491b
  ▸ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 13
    Additional RRs: 2
  ▾ Queries
    ▸ 15Ni6pff01.blr.coednssecurity.in: type TXT, class IN
  ▾ Answers
    ▾ 15Ni6pff01.blr.coednssecurity.in: type TXT, class IN
        Name: 15Ni6pff01.blr.coednssecurity.in
        Type: TXT (Text strings) (16)
        Class: IN (0x0001)
        Time to live: 299
        Data length: 5
        TXT Length: 4
        TXT: 01ND
  ▸ Authoritative nameservers
  ▸ Additional records
    [Request In: 11]
```

```
0020  02 0a 00 35 e1 10 01 39  27 14 49 1b 81 80 00 01   ···5···9 '·I·····
0030  00 01 00 0d 00 02 0a 31  35 4e 69 36 70 66 66 30   ·······1 5Ni6pff0
0040  31 03 62 6c 72 0e 63 6f  65 64 6e 73 73 65 63 75   1·blr·co ednssecu
0050  72 69 74 79 02 69 6e 00  00 10 00 01 c0 0c 00 10   rity·in· ········
0060  00 01 00 00 01 2b 00 05  04 30 31 4e 44 00 00 02   ·····+·· ·01ND···
0070  00 01 00 00 1c 8f 00 14  01 61 0c 72 6f 6f 74 2d   ········ ·a·root-
0080  73 65 72 76 65 72 73 03  6e 65 74 00 00 00 02 00   servers· net·····
0090  01 00 00 1c 8f 00 04 01  64 c0 50 00 00 02 00 01   ········ d·P·····
00a0  00 00 1c 8f 00 04 01 6a  c0 50 00 00 02 00 01 00   ·······j ·P······
```

Figure 4.5 A Network Packet Capture for ThunderDNS Traffic

the tunnel: 0<random string with 7 characters><client ID>.<domain name>.

2. Once the initial negotiation completes, the client pools data from the server as follows: 1<random string with 7 characters><client ID>.<domain name>.

3. If new data is available for the client, then the server reply with TXT records with the data; else, it will return as <id>ND.

Following signature captures this replay and confirm the ThunderDNS tunnel traffic:

IDS signature for ThunderDNS

alert udp any 53 -> any any (msg:"ThunderDNS Tunnel Traffic"; content:"|00 01 00 01|", offset 4, depth 4; content:"|00 10 00 01|", distance 4; content:"ND", distance 4, within 255; threshold: type threshold, track by_src, count 15, seconds 2;sid:806254380;rev:1;)

The above signature has three main patterns to identify ThunderDNS traffic; the first pattern is for the question and answers section as seen in the previous signatures, the second is TXT record IN class, and the third pattern is "ND," the patterns need to match 15 times in 2 seconds.

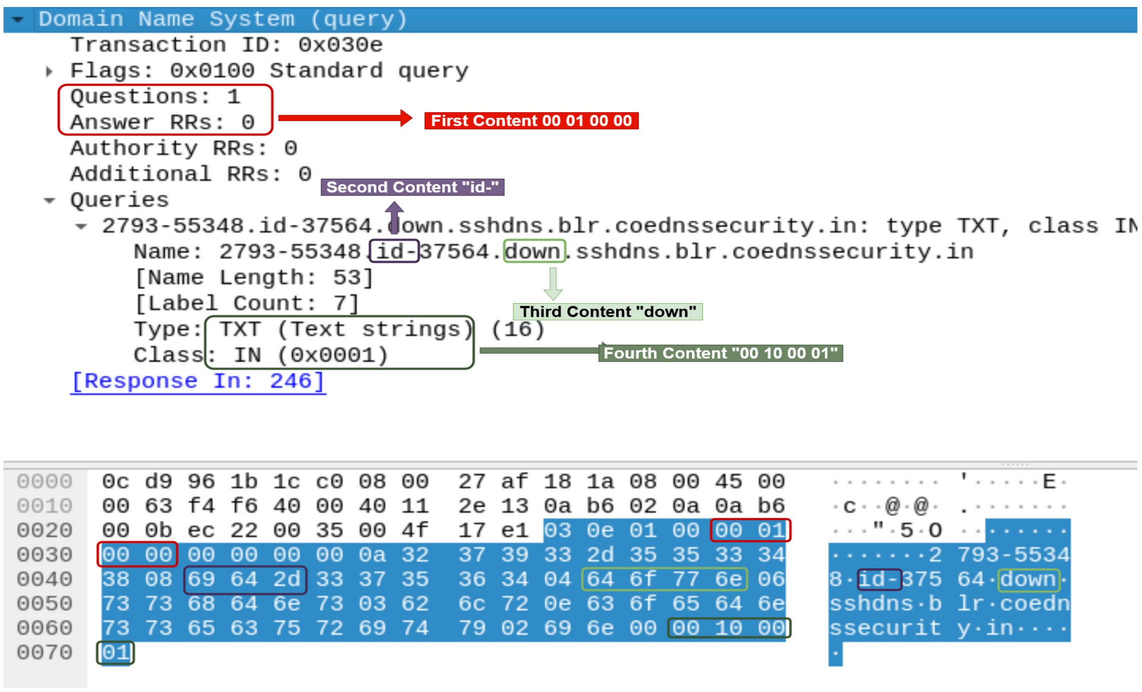
4.2.1.6 IDS signature for OzymanDNS

In 2004, Dan Kaminsky developed OzymanDNS application in Perl to create an SSH tunnel through DNS requests and responses to transmit files from client to server for data exfiltration. The DNS quires are base32 encoded, while the DNS responses are TXT records that are base64 encoded. During the tunnel negotiation, the term "up" or "down" is used in the Perl script to indicate whether the traffic is upstream or downstream from the client. As a result, we discovered two IDS signatures for OzymanDNS tunnel traffic.
Signature-1: Figure 4.6. shows the tunnel traffic captured for DNS query containing down request and based on this following signatures are created for detecting OzymanDNS tunnel traffic:

IDS signature for OzymanDNS

```
alert udp any any -> any 53 (msg:"OzymanDNS Client Down Request"; content:"|00 01 00 00|",offset 4,depth 4; content:"id-",distance 4; content:"down"; within:10; content:"|00 10 00 01|"; distance:5; within:255; threshold: type threshold, track by_src, count 20, seconds 5;sid:806254381;rev:0;)
```

The signature mentioned above has four parts: the first part (content |00 01 00 00|) corresponds to a DNS query with one question and zero answers, the second part is the "id-" keyword, the third part is the "down" keyword, and the last part (content:”|00 10 00 01|”) corresponds to a TEXT record in the IN class. In 5 seconds, this pattern must match 20 times.
Similarly following signatures detects OzymanDNS tunnel traffic for up request from client.



```
▾ Domain Name System (query)
    Transaction ID: 0x030e
  ▸ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▾ 2793-55348.id-37564.down.sshdns.blr.coednssecurity.in: type TXT, class IN
        Name: 2793-55348.id-37564.down.sshdns.blr.coednssecurity.in
        [Name Length: 53]
        [Label Count: 7]
        Type: TXT (Text strings) (16)
        Class: IN (0x0001)
    [Response In: 246]

0000  0c d9 96 1b 1c c0 08 00  27 af 18 1a 08 00 45 00   ········ '·····E·
0010  00 63 f4 f6 40 00 40 11  2e 13 0a b6 02 0a 0a b6   ·c··@·@· .·······
0020  00 0b ec 22 00 35 00 4f  17 e1 03 0e 01 00 00 01   ···"·5·O ········
0030  00 00 00 00 00 00 0a 32  37 39 33 2d 35 35 33 34   ·······2 793-5534
0040  38 08 69 64 2d 33 37 35  36 34 04 64 6f 77 6e 06   8·id-375 64·down·
0050  73 73 68 64 6e 73 03 62  6c 72 0e 63 6f 65 64 6e   sshdns·b lr·coedn
0060  73 73 65 63 75 72 69 74  79 02 69 6e 00 00 10 00   ssecurit y·in····
0070  01                                                 ·
```

Figure 4.6 A Network Packet Capture for OzymanDNS Traffic

IDS signature for OzymanDNS

alert udp any any -> any 53 (msg:"OzymanDNS Client up Request"; content:"|00 01 00 00|";offset:4;depth:4; content:"-0"; distance:16; content:"id-";distance:1;within:3; content:"up"; within:8; content:"|00 01 00 01|"; distance:5; within:255; threshold: type threshold, track by_src, count 20, seconds 5;sid: 806254382;rev:0;)

4.2.2 DNS Amplification

DNS amplification attacks are also known as reflective distributed denial of service (RDDoS) attacks. They can take two forms: being the target or acting as a middleman in attacking another network. A decent amplification attack vector meets two criteria:
1. The query can be sent from a faked source address (e.g., using an ICMP or UDP protocol that does not require a handshake)
2. The response to the query is multiple factors and much larger than the query itself.
Because DNS is a key and omnipresent Internet infrastructure that fits these conditions,

it has become the most significant amplification source. DNS is an application-layer protocol primarily used with UDP connection, and UDP is a stateless protocol that is essentially a best-effort protocol. Because there is no connection handshake, spoofing the source IP address in receiving packets is quite straightforward. According to RFC 1034 and 1035, the payload size of DNS request packets was previously limited to 512 bytes. However, due to the development of Extension Mechanisms for DNS (EDNS(0)) and later adoption of these standards, the DNS payload packet size can now reach up to 4096 bytes. A common DNS query is between 50 and 100 bytes in size, and DNS response can range in size from a few hundred bytes for an A or CNAME record to thousands of bytes or more for a big ANY or TXT record. The DNS amplification attack takes leverages of this and EDNS(0) to enable an adversary to amplify the amount of bandwidth they can target at a possible victim for a DDoS attack.

DNS Amplification employs the reflection approach, in which the attacker spoofs the

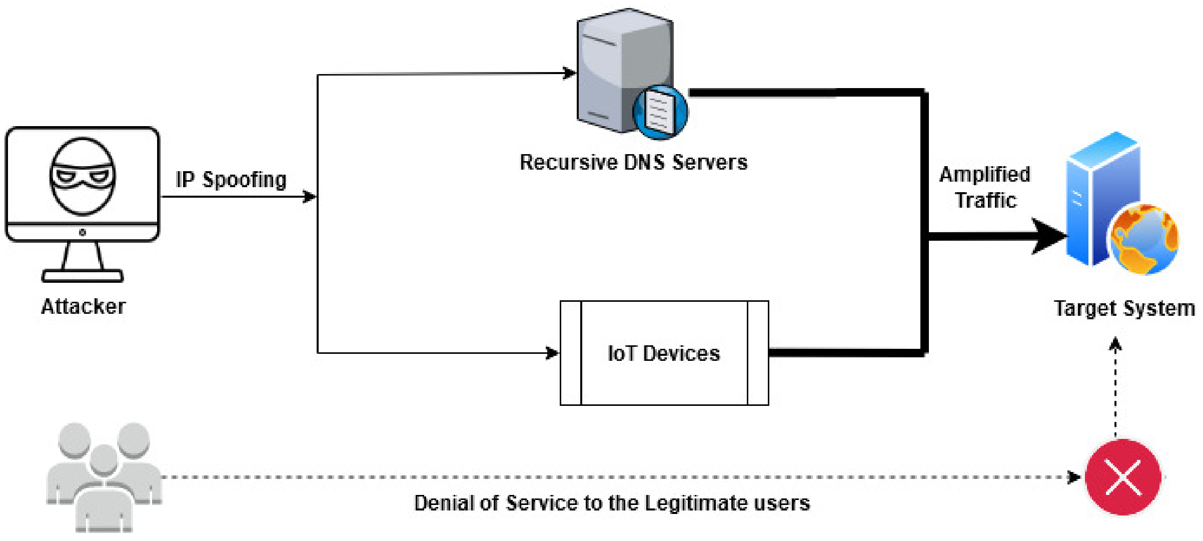


Figure 4.7 Working of DNS Amplification Attack

target system's IP address and sends a DNS query to a list of public recursive resolvers. The target system will receive a response from all the recursive resolvers in response to the query sent by the attacker. By asking for all records (ANY records) of a specified domain and enabling EDNS(0) protocol, the attacker crafts DNS query so that the amplified responses overburden the victim's network resulting DDoS attack as shown

Table 4.2 List of DNS Reflected Amplification Tools

Tool Name	Query Type	Program	Class	EDNS Support
Ethanwilloner	ANY	C	IN	No
Saddam	ANY	Python	ANY	Yes
DNS FlooderV1.1	ANY	C	IN	Yes
DNSDRDOS	A	C	IN	No

in Figure 4.7. DNS answers from authentic DNS servers are interpreted as authentic DNS communication by the victim machine; as a result, preventing these attacks is quite tough. Compared to the number of UDP payload bytes that an amplifier delivers to respond to a request, the bandwidth amplification factor (BAF) is computed as the amount of UDP payload bytes that can be used to assess the potential impact of an amplification attack and can weigh the potential outcomes of an amplification attack. The BAF is determined as the difference between the number of payloads an amplifier transmits to a victim and the number of payloads an attacker sends to an amplifier [88]. The recursive resolver's BAF ranges from 28 to 54. The following formula computes BAF:

$$\text{Bandwidth Amplification Factor (BAF)} = \frac{\text{Packet Size of DNS Response}}{\text{Packet Size of DNS Query}}$$

A selection of the most popular tools for launching DNS reflective amplification attacks is shown in the Table 4.2. All tools operate similarly, requiring a list of recursive servers and the victim system's IP address and a list of domains used in DNS queries with ANY option selected. We developed IDS signature for all these tools as illustrated in the following sections.

4.2.2.1 IDS Signature for Ethanwilloner

Ethan Willoner created a C program that may be used to demonstrate the notion of a DNS amplification attack [89]. The utility sends DNS queries to a list of open DNS servers specified in the C program, with the source spoofing the target. Because the code requires access to raw sockets, it must be run with root privileges. This tool can spoof the sender's IP address, and the source code can be modified to set recursive resolvers and domain names for DNS queries.

We captured the attack traffic as shown in Figure 4.8 and analyzed the traffic to create an IDS signature that identifies this attack pattern. The following are the essential factors for detecting this attack on the victim:

1: Standard Query Response –The DNS response from the recursive server always contains the content "81" corresponds to the Response with recursion desired, or content "83" corresponds to the response with truncated and recursion desired.

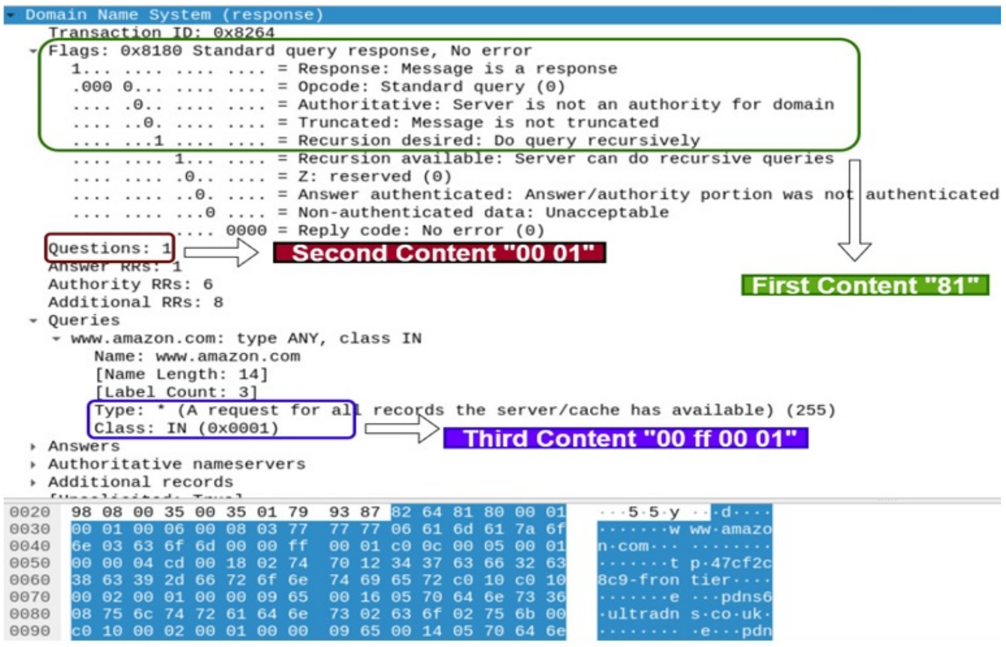
2: Question Section – The question section corresponds to the following content in the DNS header:"|00 01|".

3: Query Type – This tool uses "ANY" in query type with Internet class "IN," which is represented by content:"|00 ff 00 01|"

4: The DNS response size – We generate the alert if the DNS packet size is more than 100.

5: Threshold value – We generate an alert if 10000 packets match the signature in ten seconds.

The signatures for identifying Ethanwillnor traffic is as follows:



```
Domain Name System (response)
    Transaction ID: 0x8264
  Flags: 0x8180 Standard query response, No error
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative: Server is not an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 1... .... = Recursion available: Server can do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 1
    Authority RRs: 6
    Additional RRs: 8
  Queries
    www.amazon.com: type ANY, class IN
        Name: www.amazon.com
        [Name Length: 14]
        [Label Count: 3]
        Type: * (A request for all records the server/cache has available) (255)
        Class: IN (0x0001)
  Answers
  Authoritative nameservers
  Additional records

0020  98 08 00 35 00 35 01 79  93 87 82 64 81 80 00 01   ···5·5·y ···d····
0030  00 01 00 06 00 08 03 77  77 77 06 61 6d 61 7a 6f   ·······w ww·amazo
0040  6e 03 63 6f 6d 00 00 ff  00 01 c0 0c 00 05 00 01   n·com··· ········
0050  00 00 04 cd 00 18 02 74  70 12 34 37 63 66 32 63   ·······t p·47cf2c
0060  38 63 39 2d 66 72 6f 6e  74 69 65 72 c0 10 c0 10   8c9-fron tier····
0070  00 02 00 01 00 00 09 65  00 16 05 70 64 6e 73 36   ·······e ···pdns6
0080  08 75 6c 74 72 61 64 6e  73 02 63 6f 02 75 6b 00   ·ultradn s·co·uk·
0090  c0 10 00 02 00 01 00 00  09 65 00 14 05 70 64 6e   ········ ·e···pdn
```

Figure 4.8 Packet Capture for EthanWillnor Traffic

IDS signature for Ethanwilloner tarffic

```
alert udp any 53 -> any any (msg:"EthanWillonr-DNS Amplification Attack"; content:"|81|",offset 2,depth 1; content:"|00 01|", distance 1, within 2;content:"|00 ff 00 01|",distance 6, within 100; dsize:>100; threshold: type threshold, track by_src, count 10000, seconds 10; sid:806254383;rev:1;)
alert udp any 53 -> any any (msg:"EthanWillonr-DNS Amplification Attack"; content:"|83|",offset 2,depth 1; content:"|00 01|", distance 1, within 2;content:"|00 ff 00 01|",distance 6, within 100; dsize:>100; threshold: type threshold, track by_src, count 10000, seconds 10; sid:806254384;rev:1;)
```

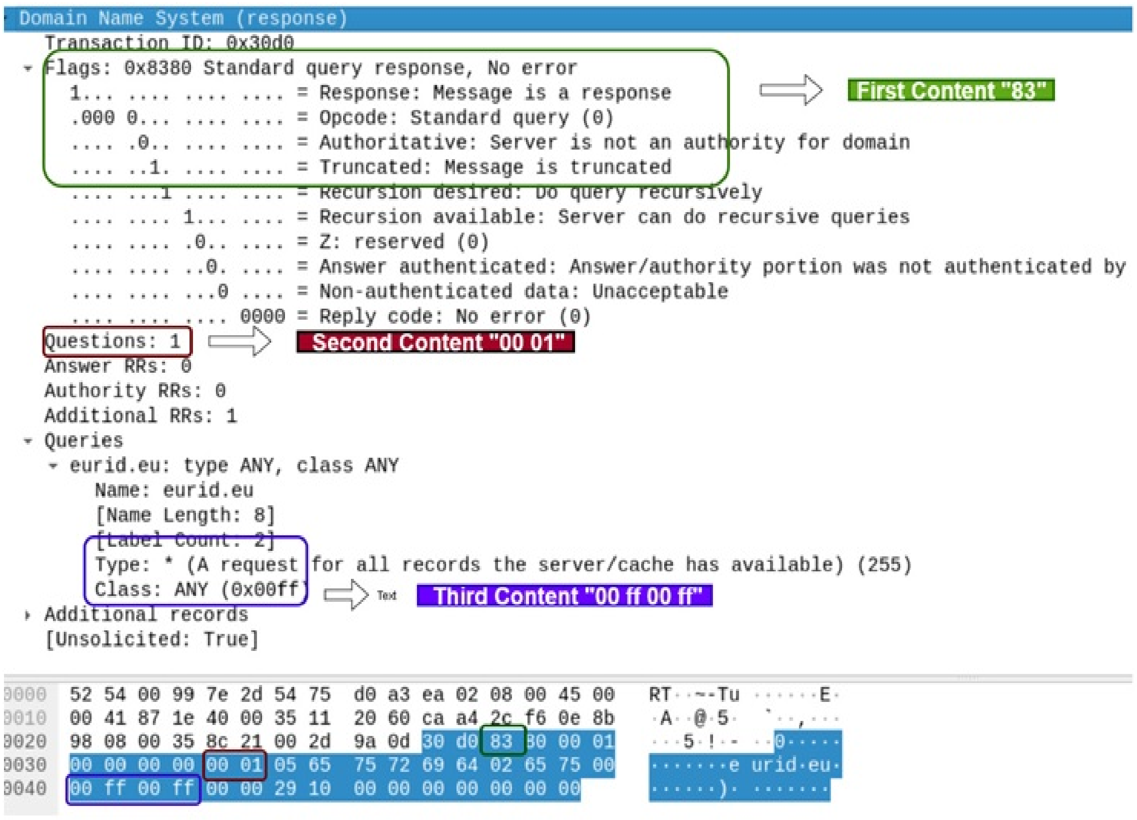
4.2.2.2 IDS signature for Offensive-Python Saddam

The offensive python Saddam utility [90] is a python application that may execute different amplification attacks, such as DNS, SNMP, NTP, and SSDP. This program needed raw socket functionality on the operating system as well as the Python 2.7 Pinject module. It allows for benchmarking and exposes the amplification factor for a collection of open recursive resolvers.

IDS signature for OffensivePython-Saddam tarffic

```
alert udp any 53 -> any any (msg:"Saddam-DNS Amplification Attack"; content:"|81|", offset 2, depth 2; content:"|00 01|", distance 1, within 2;content:"|00 ff 00 ff|", distance 8; threshold: type threshold, track by_src, count 10000, seconds 10; sid:806254385;rev:1;)
alert udp any 53 -> any any (msg:"Saddam-DNS Amplification Attack"; content:"|83|", offset 2, depth 2; content:"|00 01|", distance 1, within 2;content:"|00 ff 00 ff|", distance 8; threshold: type threshold, track by_src, count 10000, seconds 10; sid:806254385;rev:1;)
```

If the sadam tool is used for an amplification attack, the above signature will raise an alert. Except for the query type, the parameters are identical to those of the ethanwilloner



```
Domain Name System (response)
    Transaction ID: 0x30d0
  ▾ Flags: 0x8380 Standard query response, No error
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative: Server is not an authority for domain
      .... ..1. .... .... = Truncated: Message is truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 1... .... = Recursion available: Server can do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 1
  ▾ Queries
    ▾ eurid.eu: type ANY, class ANY
        Name: eurid.eu
        [Name Length: 8]
        [Label Count: 2]
        Type: * (A request for all records the server/cache has available) (255)
        Class: ANY (0x00ff)
  ▸ Additional records
    [Unsolicited: True]

0000  52 54 00 99 7e 2d 54 75  d0 a3 ea 02 08 00 45 00   RT··~-Tu ······E·
0010  00 41 87 1e 40 00 35 11  20 60 ca a4 2c f6 0e 8b   ·A··@·5· `··,···
0020  98 08 00 35 8c 21 00 2d  9a 0d 30 d0 83 80 00 01   ···5·!·- ··0·····
0030  00 00 00 00 00 01 05 65  75 72 69 64 02 65 75 00   ·······e urid·eu·
0040  00 ff 00 ff 00 00 29 10  00 00 00 00 00 00 00      ······)· ·······
```

Figure 4.9 Packet Capture for Offensive-Python Saddam Tooltik Traffic

tool. Any query type will be applied with any class represented by content:"|00 ff 00 ff|".

Figure 4.9 shows a network capture of Offensive-Python Saddam traffic.

4.2.2.3 IDS Signature for DNS Flooder-v1.1

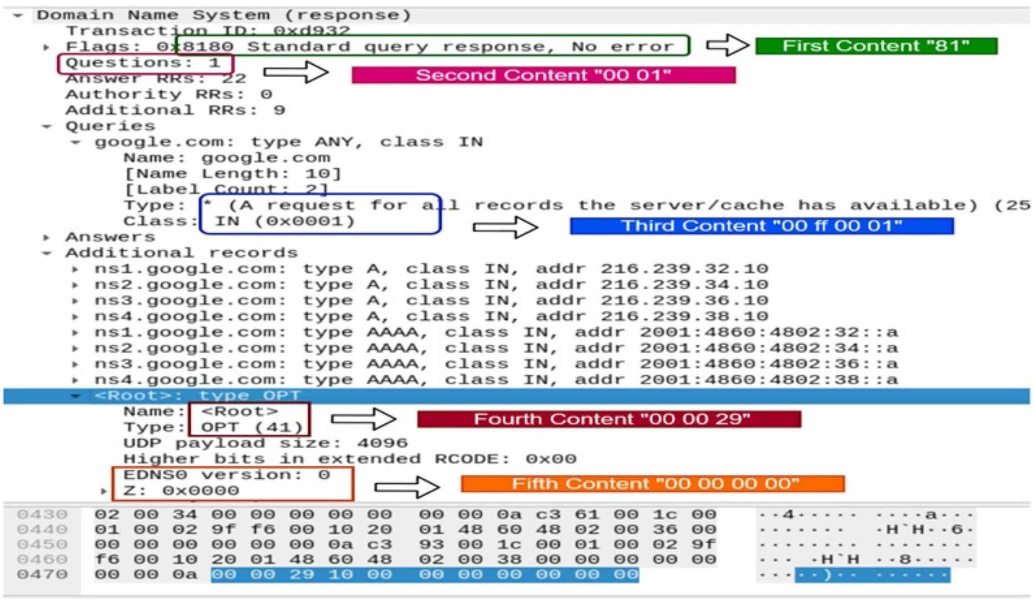
Prolexic discovered this toolkit threat in 2013, written in C, and it features a new, popular technique of creating big DNS resource records for the response of DNS queries [91]. With this, malicious actors can now magnify replies by a factor of 50 or more. This toolkit includes a different record section that uses EDNS(0) to increase the UDP response size so that the server responds with the largest feasible response. The tool uses ANY request with a spoofed IP address for performing a reflected amplification attack. We captured DNS Flooder-1.1 attack traffic as shown in Figure 4.10, and identified the following signature to detect DNS flooder traffic:

IDS signature for DNS Flooder v1.1 tarffic

alert udp any 53 -> any any (msg:"DDoS Attack attempt using DNS flooder 1.1";content:"|81|",offset 2,depth 2; content:"|00 01|",distance 1, within 2; content:"|00 ff 00 01|",distance 8; content: "|00 00 29|",distance 50;content:"|00 00 00 00|",distance 2, within 4;dsize:>200;threshold: type threshold, track by_src, count 10000, seconds 10; sid:9999993;rev:1;)

alert udp any 53 -> any any (msg:"DDoS Attack attempt using DNS flooder 1.1";content:"|83|",offset 2,depth 2; content:"|00 01|",distance 1, within 2; content:"|00 ff 00 01|",distance 8; content: "|00 00 29|",distance 50;content:"|00 00 00 00|",distance 2, within 4;dsize:>200;threshold: type threshold, track by_src, count 10000, seconds 10; sid:9999993;rev:1;)

The signature is the same as Ethanwalliner except for EDNS(0), The content “|00 00 29|” corresponds to OPT query type in an additional section of the response, and content “|00 00 00 00| corresponds to Z bits.



```
▾ Domain Name System (response)
    Transaction ID: 0xd932
  ▸ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 22
    Authority RRs: 0
    Additional RRs: 9
  ▾ Queries
    ▾ google.com: type ANY, class IN
        Name: google.com
        [Name Length: 10]
        [Label Count: 2]
        Type: * (A request for all records the server/cache has available) (25
        Class: IN (0x0001)
  ▸ Answers
  ▾ Additional records
    ▸ ns1.google.com: type A, class IN, addr 216.239.32.10
    ▸ ns2.google.com: type A, class IN, addr 216.239.34.10
    ▸ ns3.google.com: type A, class IN, addr 216.239.36.10
    ▸ ns4.google.com: type A, class IN, addr 216.239.38.10
    ▸ ns1.google.com: type AAAA, class IN, addr 2001:4860:4802:32::a
    ▸ ns2.google.com: type AAAA, class IN, addr 2001:4860:4802:34::a
    ▸ ns3.google.com: type AAAA, class IN, addr 2001:4860:4802:36::a
    ▸ ns4.google.com: type AAAA, class IN, addr 2001:4860:4802:38::a
    ▾ <Root>: type OPT
        Name: <Root>
        Type: OPT (41)
        UDP payload size: 4096
        Higher bits in extended RCODE: 0x00
        EDNS0 version: 0
      ▸ Z: 0x0000
0430  02 00 34 00 00 00 00 00  00 00 0a c3 61 00 1c 00   ··4·····  ····a···
0440  01 00 02 9f f6 00 10 20  01 48 60 48 02 00 36 00   ········  ·H`H··6·
0450  00 00 00 00 00 00 0a c3  93 00 1c 00 01 00 02 9f   ········  ········
0460  f6 00 10 20 01 48 60 48  02 00 38 00 00 00 00 00   ··· ·H`H  ··8·····
0470  00 00 0a 00 00 29 10 00  00 00 00 00 00 00         ·····)··  ······
```

First Content "81"
Second Content "00 01"
Third Content "00 ff 00 01"
Fourth Content "00 00 29"
Fifth Content "00 00 00 00"

Figure 4.10 Packet Capture for DNS Flooder-v1.1 Tarffic

4.2.2.4 IDS Signature for DNSDRDOS

In 2015, Nullsecurity created a C programming code called "dnsdrdos.c" used as a proof-of-concept for demonstrating a widespread DNS reflection attack that results in a denial of service on the victim's network [92]. The code must be built to generate an object code executable that requires a list of open recursive resolvers, victim IP address, a domain name to resolve, and a loop count. The adversary may successfully conduct a DDoS attack from several hosts, utilizing multiple open name servers, and overwhelm the target with undesired network traffic.
By analysing wireshark traffic shown in Figure 4.11, the ensuing signatures were created to detect DNSDRDOS traffic at the victim as:

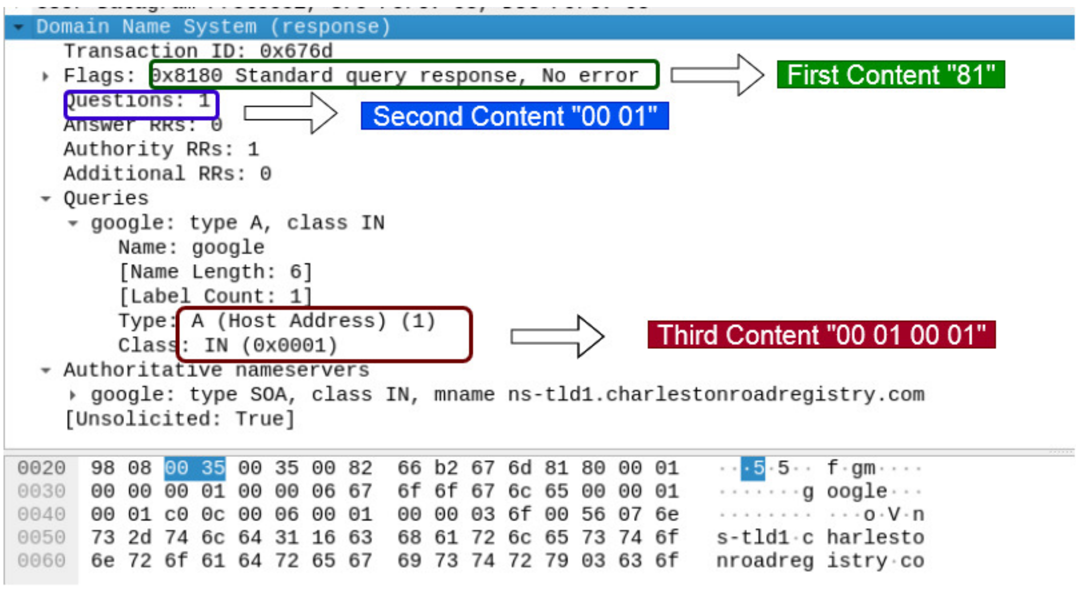
IDS Signature for DNSDRDOS Tarffic

```
alert udp any 53 -> any any (msg:"DDoS Attack attempt using DNDDRDOS.C
Program"; content:"|81|",offset 2,depth 1; content:"|00 01|", distance 1, within
2;content:"|00 01 00 01|",distance 6; dsize:>100; threshold: type threshold, track
by_src, count 10000, seconds 10; sid:806254388;rev:1;)
alert udp any 53 -> any any (msg:"DDoS Attack attempt using DNDDRDOS.C
Program"; content:"|83|",offset 2,depth 1; content:"|00 01|", distance 1, within
2;content:"|00 01 00 01|",distance 6; dsize:>100; threshold: type threshold, track
by_src, count 10000, seconds 10; sid:806254388;rev:1;)
```

The IDS signatures are similar to the other amplification attack tools. However, for DNSDRDOS, we observed that the response size is always greater than 100 bytes, and 10000 similar packets were found every 10 seconds to identify this attack.

4.2.3 IDS Signature for DoS Attack on DNS Servers

An attacker can sometimes target DNS servers by flooding a particular domain's authoritative server or recursive resolver to disrupt the DNS resolution process, causing a DoS attack on DNS called DNS flood. In this case, the attacker tries to flood a DNS server



```
▾ Domain Name System (response)
    Transaction ID: 0x676d
  ▸ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 0
    Authority RRs: 1
    Additional RRs: 0
  ▾ Queries
    ▾ google: type A, class IN
        Name: google
        [Name Length: 6]
        [Label Count: 1]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
  ▾ Authoritative nameservers
    ▸ google: type SOA, class IN, mname ns-tld1.charlestonroadregistry.com
    [Unsolicited: True]

0020  98 08 00 35 00 35 00 82  66 b2 67 6d 81 80 00 01   ···5·5·· f·gm····
0030  00 00 00 01 00 00 06 67  6f 6f 67 6c 65 00 00 01   ·······g oogle···
0040  00 01 c0 0c 00 06 00 01  00 00 03 6f 00 56 07 6e   ········ ···o·V·n
0050  73 2d 74 6c 64 31 16 63  68 61 72 6c 65 73 74 6f   s-tld1·c harlesto
0060  6e 72 6f 61 64 72 65 67  69 73 74 72 79 03 63 6f   nroadreg istry·co
```

Figure 4.11 Packet Capture for DNSDRDOS Tarffic

with ostensibly valid traffic, overloading server resources and hindering the server's ability to direct genuine requests to domain resources. Therefore, a DNS flood attack compromises the ability of a website, API, or any online application to react by interrupting DNS resolution. As depicted in Figure 4.12, several controlled systems known as bots flood the target DNS server with DNS requests, rendering it unavailable to legitimate users for DNS resolution resulting in the Distributed Denial of Service (DDoS) attack. Because the huge volume of traffic typically originates from various places, searching for correct records on the domain, and imitating legitimate traffic, DNS flood assaults can be difficult to differentiate from regular heavy traffic. DNS flood attacks have been affecting root name servers for almost 17 years and have been on the rise recently. In order to overrun the name server, DNS flood attacks frequently employ high-bandwidth IoT devices such as NVR devices, IP cameras, Canary, Ring doorbells, etc., which become part of botnets such as Mirai. A number of applications may be used to create normal DNS requests with the faked/spoofed IP address and send them to targeted DNS servers for DoS. DNS flooding programs such as Google's DNS-Flood [93] and Python's SCAPY are popular for crafting DNS queries.

When DNS servers receive more than 10000 DNS queries per ten seconds, the follow-

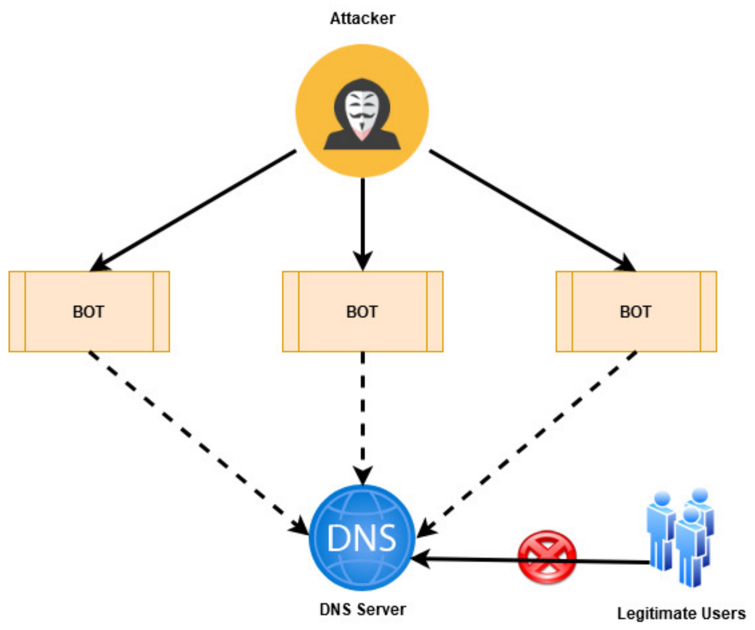


Figure 4.12 DDoS Attacks on DNS Servers by using DNS Flood.

ing rule will alert "DNS Flood Attempt" :

IDS Signature for DoS Attack on DNS

```
alert udp any any -> any 53 (msg:"DNS Flooder-DoS Attack"; content:"|01 00 00 01|", offset 2, depth 4; content:"|00 01|", distance 10, within 255;threshold: type threshold, track by_src, count 10000, seconds 10; sid:806254390;rev:1;)
```

4.3 Comparative Evaluation of DID and SNORT Concerning DNS Attacks

SNORT IDS has more than 25000 signatures to detect anomalies in the network traffic, and among them, there are very few signatures to detect DNS-based anomalies. The snort signatures are signatures for TCP and UDP-based DoS and DDoS attacks, attacks on various applications like FTP, TELNET, SSH and DNS, etc., attacks on DBMS, web application attacks, and attacks on e-mail related protocols like POP, IMAP, & SMTP. SNORT contains several predefined signatures to defend DNS from different attacks, and these existing signatures can protect from "Attempt on DNS name and version" and "Attempt on zone transfer." Two files, “dns.rules” and “protocol-dns,” contain DNS-related signatures but lacks signature for some of the tools described in the section-4.2

for DNS tunneling and amplification attacks. We created a separate set of signatures for detecting DNS-based abuses, including existing signatures related to DNS attacks and novel signatures developed in previous sections. SNORT is then used in conjunction with the resulting signature set to detect intrusion in DNS infrastructure, and this solution is known as "DNS Intrusion Detection - DID."

To the best of our knowledge and when writing this thesis, there is no publicly accessible produced dataset is available for all DNS attacks. As a result, we conducted experiments in a real environment to produce the dataset we needed for our research. The experiment setup is made for the evaluation of DID and SNORT signatures in a controlled but real environment; we created an Internet domain name "blr.coednssecurity.in" for performing DNS tunneling attacks, and we also used our public DNS resolvers 223.31.121.171 and 14.139.152.4 as DNS reflector to simulate amplification attacks in order to avoid legal issues. We chose a physical server with 128 GB of RAM and 16 core two sockets CPU for our experiment. This server is configured for virtualization with the widely used hypervisor "KVM" on CentOS 8.1. In this experiment, we installed DID in a virtual computer running CentOS 8.1. The victim system is running on a different CentOS8.1 virtual machine. As an attack node, Kali Linux virtual machine is set up on another physical server on a separate physical network than the target system. It is used to perform DNS-based attacks on the victim system. As shown in the Figure 4.13, the following are the components of the experiment setup we established:

1. Target Node : In the experiment, this node is targeted by the attack node; in the case of tunneling attacks, the client component of each tunneling tool is installed on this node which creates a DNS tunnel with the attack node; and in the event of amplification attacks, attack node spoofs the IP address of this system and sends DNS request to our public DNS resolvers using various amplification attack tools, resulting in massive DNS response traffic to this system.

2. Attack Node : This node is used to carry out various attacks, including DNS tunneling, amplification, and DNS Flood attacks, utilizing all of the tools described in section-4.2.

3. SNORT IDS : With about 25000 registered signatures, this node is equipped with

Table 4.3 Experiment Setup Configurations

System Name	CPU assigned	Memory Assigned	OS used	IP address Assigned
Victim System	8 Core Xeon	16 GB	Centos 8.1	14.139.152.14
DID	8 Core Xeon	16 GB	Centos 8.1	14.139.152.5
SNORT-IDS	8 Core Xeon	16 GB	Centos 8.1	14.139.152.6
Attacker System	8 Core Xeon	32 GB	Kali 2020	223.31.121.172

SNORT version 2.9, all of the gateway's network traffic is mirrored on this node to get a DNS anomaly.

4. DID: This node has SNORT version 2.9 installed and configured, and except for DNS rules, all signatures are deleted, and the signatures produced in section-4.2 are appended to the existing DNS rules. Like the SNORT IDS node, this node receives all network traffic from the gateway through switch port mirroring.

Throughout the experiment, the target node generated legitimate DNS requests to different DNS servers. Table 4.3 lists the configuration of each system in the experimental setup. The confusion matrix, which reflects the outcome of categorization in true and

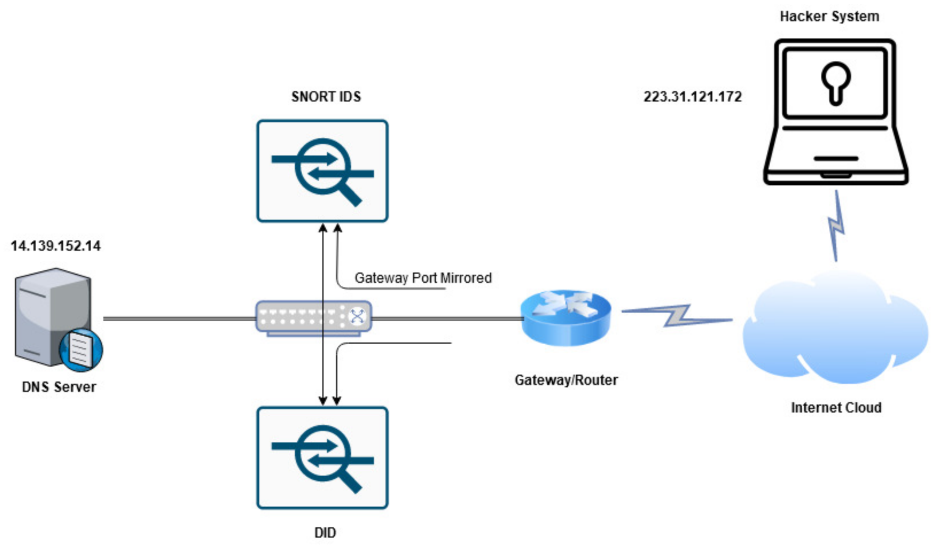


Figure 4.13 Experiment Setup for Evaluation of DID and SNORT

false [94], is a matrix used to evaluate the performance of IDS. As indicated in Ta-

Table 4.4 Confusion Matrix

Actual	Predicted	
	Legitimate	Intrusion
Legitimate	True Negative	False Positive
Intrusion	False Negative	True Positive

ble 4.4, there are several options for categorizing events.

True Positive : Successfully recognized intrusions by the IDS.

True Negative : The IDS detects valid traffic and marks it as such.

False Positive : The IDS incorrectly classifies legitimate traffic as intrusive traffic.

False Negatives : These are intrusions that the IDS misses and labels as legitimate traffic. We use standard performance metrics to evaluate IDS, and the following are the standard performance measures:

Detection Rate (or Sensitivity) : This is the ratio of expected attacks to all attacks. It is exceedingly unusual for an IDS to properly detect all potential assaults, resulting in a ratio of 1. True positive rate is another name for it, and it may be represented mathematically as:

$$\text{True Positive Rate (TPR)} = \frac{TP}{TP+FN}$$

False Positive Rate : The ratio of the number of valid cases incorrectly categorized as an intrusion to the total number of legitimate instances is known as the False Positive Rate and represented mathematically as:

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP+TN}$$

Accuracy : The accuracy, often known as the classification rate, is computed as the proportion of properly categorized cases to the total number of occurrences. It assesses an IDS's ability to identify genuine or malicious traffic. Accuracy called as the fraction of data categorised as the proper kind in the entire data. Real situations are True Positive (TP) and True Negative (TN), while false detected situations are False Positive (FP) and False Negative (FN). The following equation calculates the accuracy of the system:

$$Accuracy(CR) = \frac{TP+TN}{TP+TN+FP+FN}$$

To the best of our knowledge, there are no datasets that simulate DNS-based assaults when writing this work. As a result, the experiment setup was conducted for 22 hours, during which we monitored DNS activity and classified it as genuine or intrusion.

The effectiveness of DID signatures was assessed using the following experimental attacks:

Experiment-1 : The DNS amplification attacks were carried out using attack nodes connected to different ISP networks then target nodes. The attacking node used all DNS amplification attack tools mentioned in section-4.2 to initiate the attacks against different instances in 22 hours.

Experiment-2 : DNS Tunnelling attacks were performed as part of this experiment utilizing the DNS tunneling attack tools mentioned in section-4.2. In 22 hours, the attack was launched many times.

Experiment-3 : For 22 hours, the DoS attack was carried out from attack node to victim node running an authoritative DNS server, utilizing various tools such as DNS-Flood and Hping on distinct instances.

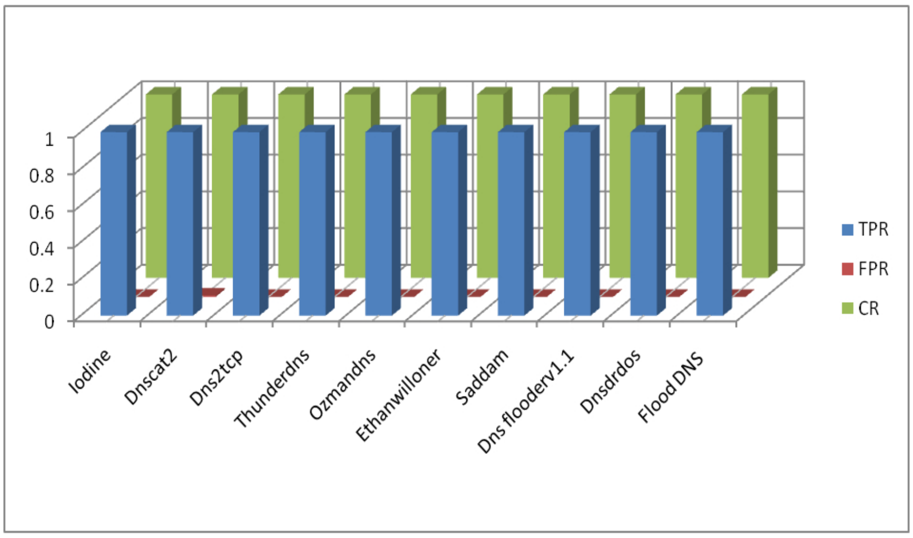


Figure 4.14 Performance Analysis of DID

Table 4.5 Evaluation of SNORT and DID Signatures

Attack Type	Attack tool Name	Number of Instances	Number of Alerts	
			SNORT	DID
DNS tunnel	Iodine	4	0	4
	Dnscat2	2572	0	3610
	Dns2tcp	8	0	8
	Thunderdns	11	0	11
	Ozmandns	277	0	277
DNS Amplification	Ethanwilloner	1070000	0	107
	Saddam	18150000	0	1815
	Dns flooderv1.1	140000	0	14
	Dnsdrdos	21030000	0	2103
DoS	Flood DNS	890000	0	89
Legitimate Traffic (True Negative)		**5408922**		

Table 4.6 Efficiency of DID Signatures

Tool Name	True Positive	False Positive	False Negative
Iodine	4	0	0
Dnscat2	3610	1038	0
Dns2tcp	8	0	0
Thunderdns	11	0	0
Ozmandns	277	0	0
Ethanwilloner	107	0	0
Saddam	1815	0	0
Dns Flooderv1.1	14	0	0
Dnsdrdos	2103	0	0
Flood DNS [28]	89	0	0
Total	**8038**	**1038**	**0**

The efficacy of the suggested signatures in detecting various DNS-based attacks is shown in Table 4.5 and Table 4.6. Our findings demonstrate that the detection rate for DNS attacks has significantly improved. Unlike the traditional DNS defensive mechanism i.e. SNORT, the suggested new rules can accurately identify DNS amplification, DNS tunneling, and DNS-based DoS/DDoS assaults. The Figure 4.14 shows the TPR, FPR, and CR for signatures produced in section-4.2, except for signatures of DNSCAT2 (CR=0.9998), the CR is 1 for all other signatures. There are roughly 25000+ signatures in SNORT (including DNS-related signatures), but only about 100 signatures in DID, all of which are DNS-related signatures. As a result, DID outperforms SNORT when it comes to defending DNS-based attacks.

4.4 Summary

We have created novel IDS signatures for the following DNS-based attacks: DoS, Amplification/Reflection, Tunneling, and added them to the existing ruleset file of SNORT IDS to detect DNS-based intrusions.
Our approach successfully identifies empirical DNS attacks performed by various known tools available over the Internet. Evaluation of DID showed a high detection rate and a very low false-positive rate.

Chapter 5

IPv6 Aware Dual-Stack DNS Hierarchy Testbed Setup and it's Evaluation

5.1 Introduction

The importance of the DNS application for accessing the Internet and the threats it faces are widely recognized. As a result, DNS has been developed and implemented as a globally dispersed, highly scalable infrastructure resistant to a wide range of attacks. Our civilization is largely anticipated to become safer, smarter, and more sustainable due to the Internet of Things (IoT). It is expected to connect approximately 100 billion of these devices to the Internet in the next decade [95]. IoT is a chance to significantly improve the value of DNS because each of them has a unique IP address that DNS can only resolve. With the exponential increase in the number of digital devices connecting to the Internet, we would be compelled to upgrade to IPv6, hence the need for DNS over IPv6. However, the guidance required for commissioning DNS over IPv6 is widely lacking, and this research work attempts to bridge this knowledge gap.

DNS latency is a critical metric for determining how responsive a DNS server is to IoT devices and, as a result, how they perceive the speed and performance of online services. Because IoT devices require end-to-end communication and the number of Internet devices has already exceeded the IPv4 limit, it is critical to maintaining network availability and performance. Additionally, as IPv6-enabled IoT devices become more prevalent, DNS traffic will increase dramatically, necessitating DNS query latency testing.

The DNS query latency from a particular Internet vantage point for IPv4 and IPv6 cannot be compared directly due to variations in the number of hops of query on IPv4 and IPv6 communication networks. Moreover, there is no assurance that the DNS server in the hierarchy is hosted on a dual-stack. This research work aims to determine the DNS query latency difference between IPv4 & IPv6 protocols. It also gives a clear illustration and provides reference guidelines for setting up a three-level DNS hierarchy (ROOT, TLD, SLD, TTLD, and recursive resolver) on a dual IP stack (IPv4 and IPv6), enabling both forward and reverse lookup trees.

Therefore a first-of-its-kind and live testbed setup is established, tested successfully, and made available for the benefit of Internet researchers, ensuring constant hops between the recursive resolver and each of the DNS servers in the hierarchy.

IPv6 is a network layer protocol, which supports communication, and data transfer over the network. It was introduced in 1998 to replace the fast depleting IPv4 address space. IPv4 protocol, the previous standard, consists of four octacts, each byte/octet is written in a dotted decimal notations. A standard IPv4 address is 32-bit, which allows for 4.2 billion unique IP addresses. To support more unique IP addresses, IPv6 was introduced. The adoption of IPv6 is ever-increasing, and India has been leading in its availability and preference within the Asian continent.

The major improvement in IPv6 over IPv4 is that IP addresses are lengthened from 32 bits to 128 bits. IPv6 provides 340 undecillion IP addresses compared to 4.2 billion IPv4 addresses [96]. This huge number of IP addresses provided relief from the threat of the exhaustion of IP addresses to connect computers or IoT devices hosts in a network. IPv6 protocol, which is of 128-bits, consists of eight numbered strings, each containing four characters (alphanumeric), separated by a colon. As 3.4 x 10^38 addresses are available in the new 128-bit IPv6 address space [98], ensuring that we won't run out of unique IP numbers to give to new devices very soon. Auto-configuration is also supported in IPv6 to assist address the majority of the flaws in IPv4.

Over the last two decades, DNS has improved security and privacy while also expanding the types of applications it can handle. Moreover, this expansion has been delayed by the enormous installed base with a diverse variety of implementations that are reluctant to update. Due to DNS optimizations, caching, and distributed operation, changes must be carefully planned, and their impact is difficult to quantify. As the number of digital devices increasing exponentially, we are compelled to upgrade to IPv6, hence the need for DNS over IPv6.

This research work stresses and anticipates that worldwide Internet infrastructure will be updated to serve many participants by using dual-stack network protocols. To the best of our knowledge, there is no prior study that investigates the latency concerns associated with DNS queries over an IPv6 network or one that highlights the difficulties in commissioning DNS hierarchy over dual-stack. We have started by implementing an IPv6-aware DNS infrastructure. With this update, we can eliminate IPv4's shortcomings, such as restricted address space, huge routing tables, tiny packet sizes, and inflexible fixed length headers. The DNS query latency of the dual-stack DNS testbed has been tested for both IPv6 and IPv4 Internet protocols.

5.2 DNS Hierarchy Testbed Setup

The experiment aims to establish a three-level DNS hierarchy for forward lookups that resolve domain names to IP addresses and a four-level DNS hierarchy for reverse lookups that resolve IP addresses to domain names. The nodes of the hierarchy are Root, TLDs, SLDs, TTLD, and Recursive resolvers. The entire hierarchy is built on dual-stack network protocols, with each node having two network interfaces for IPv4 and IPv6 networks. CentOS7.0 Linux system is used as the base operating system in all the nodes in the hierarchy, and Bind-9.9.4-61 is used on all DNS servers in the hierarchy. The DNS hierarchy testbed setup is connected to the real Internet through public IP addresses, as shown in Table 5.1 . Figure 5.1 depicts the design of the DNS hierarchy for both forward and reverse lookup trees.

5.2.1 Commissioning of Forward Lookup DNS hierarchy

The forward lookup DNS hierarchy is used to resolve the domain name to the IP addresses; the Internet vantage point sends a DNS query asking A record for "www.coednssecurity.in" to the recursive resolver which queries the entire hierarchy for the result. The forward lookup DNS hierarchy nodes are Root DNS, TLD DNS for "in" domain, and SLD DNS

Table 5.1 List of IP Addresses used in each Node of the Testbed Hierarchy

Nodes in the Hierarchy	IPv6 Address	IPv4 Address
Root Server (.)	2405:8a00:8001::9	14.139.152.9
TLD ("in." and "arpa.") Server	2405:8a00:8001::10	14.139.152.10
SLD ("coednssecurity.in." and "IP6.ARPA.")	2405:8a00:8001::11	14.139.152.11
Subdomain ("1.0.0.8.0.0.a.8.5.0.4.2.IP6.ARPA") Server	2405:8a00:8001::12	14.139.152.12
Recursive Resolver - RR	2405:8a00:8001::13	14.139.152.13

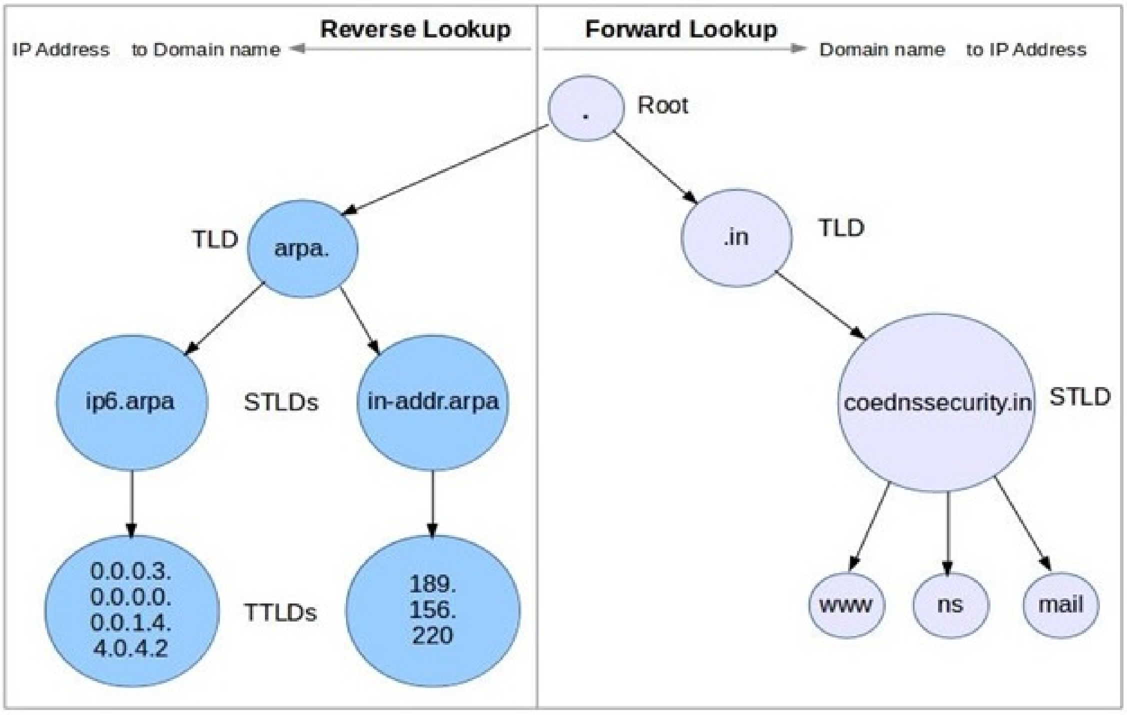
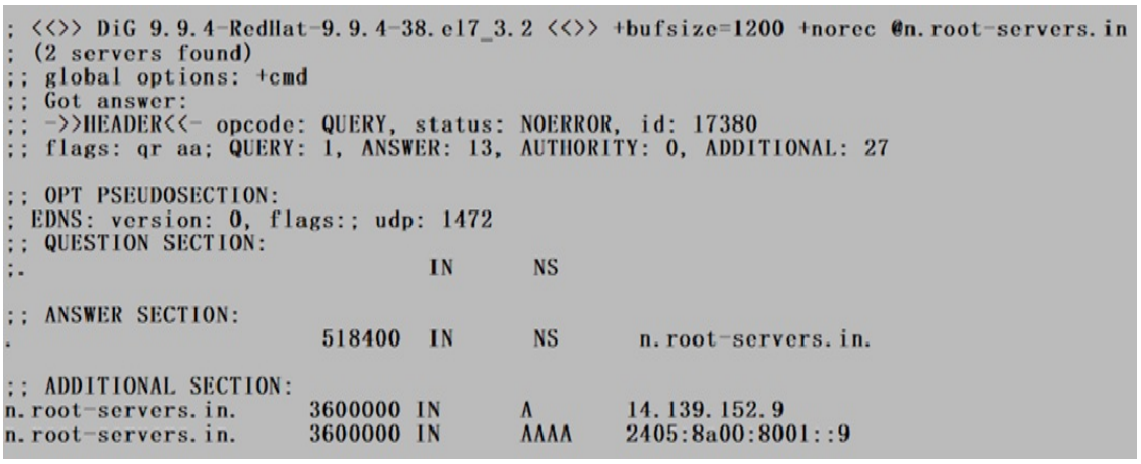


Figure 5.1 IPv6 aware DNS Tree for the Testbed

for "coednssecurity.in" domain. When the recursive resolver receives a DNS query, if it does not know the answer, it sends a query to the root server, for which it needs to know the IP address of the root. The root hints file will be part of every node that provides an authoritative server for the root domain "." as well as their IPv4 and IPv6 addresses. The root hints file "/var/named/named.ca" for all the nodes in the hierarchy is as shown Figure 5.2.



```
; <<>> DiG 9.9.4-RedHat-9.9.4-38.el7_3.2 <<>> +bufsize=1200 +norec @n.root-servers.in
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17380
;; flags: qr aa; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 27

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1472
;; QUESTION SECTION:
;.                               IN      NS

;; ANSWER SECTION:
.                       518400   IN      NS      n.root-servers.in.

;; ADDITIONAL SECTION:
n.root-servers.in.      3600000 IN      A       14.139.152.9
n.root-servers.in.      3600000 IN      AAAA    2405:8a00:8001::9
```

Figure 5.2 Root Hints File "/var/named/named.ca"

5.2.1.1 Installation and configuration of the Root server

The Domain Name System (DNS) is organized into a hierarchy of controlled regions or "zones," The root zone is at the top and DNS servers in the root zone are known as root servers. We have created a new Root DNS domain," n.root-servers.in", that contains the information of TLD domains, i.e., "in" and "arpa". Following configurations are made in bind software to create a root server:

1. bind configuration file "/etc/named.conf"

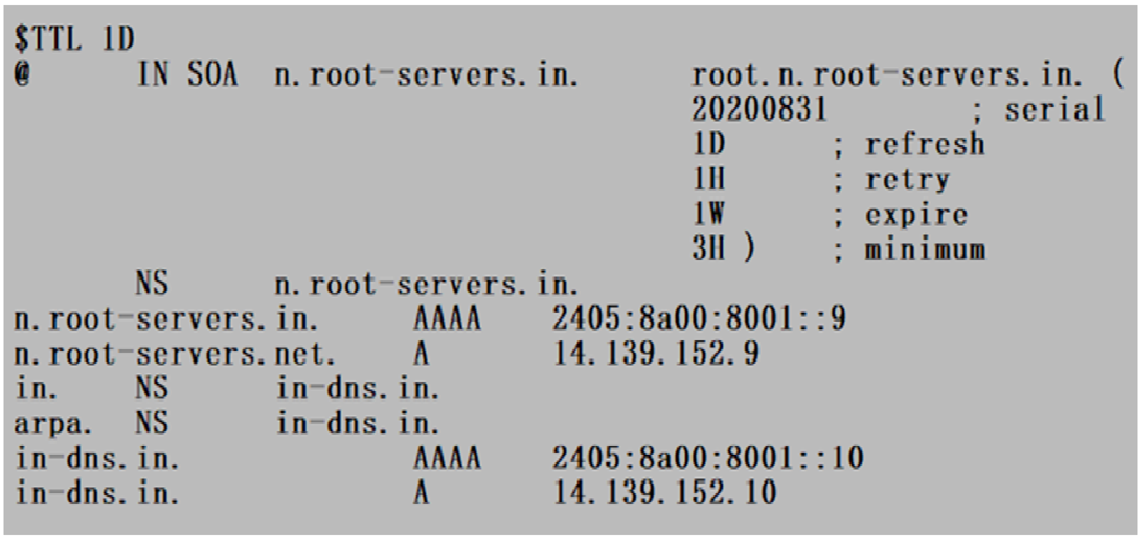
Bind configuration file "/etc/named.conf" for Root DNS

```
options { listen-on port 53 { 14.139.152.9; };
listen-on-v6 port 53 { 2405:8a00:8001::9; };
allow-query { any; };
recursion no; };
zone "." IN { type master;
file "root-zone"; };
```

Here the root name server has two network interfaces, each for IPv4 and IPv6, and it will run on both the interfaces; the root servers can be allowed to be queried from any host on the Internet. The recursion is disabled for protecting from DoS attacks, and the zone file for the root domain is "/var/named/root-zone".

2. Zone file configuration "/var/named/root-zone

The zone file needs an SOA record and NS records for the root domain. As shown in the Figure 5.3, we added delegation to the "in" zone by adding NS record "in. NS in-dns.in.". The name servers requires glue AAAA records as " in-dns.in. AAAA 2405:8a00:8001::10" and glue A records as " in-dns.in. A 14.139.152.10".



```
$TTL 1D
@      IN SOA   n.root-servers.in.      root.n.root-servers.in. (
                                        20200831        ; serial
                                        1D      ; refresh
                                        1H      ; retry
                                        1W      ; expire
                                        3H )    ; minimum
       NS       n.root-servers.in.
n.root-servers.in.      AAAA    2405:8a00:8001::9
n.root-servers.net.     A       14.139.152.9
in.    NS       in-dns.in.
arpa.  NS       in-dns.in.
in-dns.in.              AAAA    2405:8a00:8001::10
in-dns.in.              A       14.139.152.10
```

Figure 5.3 Zone File for Root DNS (/var/named/root-zone)

5.2.1.2 Installation and configuration of "in" TLD server

We configured only one TLD in the hierarchy testbed, i.e., "in" for the forward lookup tree. Bind software for the "in" domain is configured as follows:

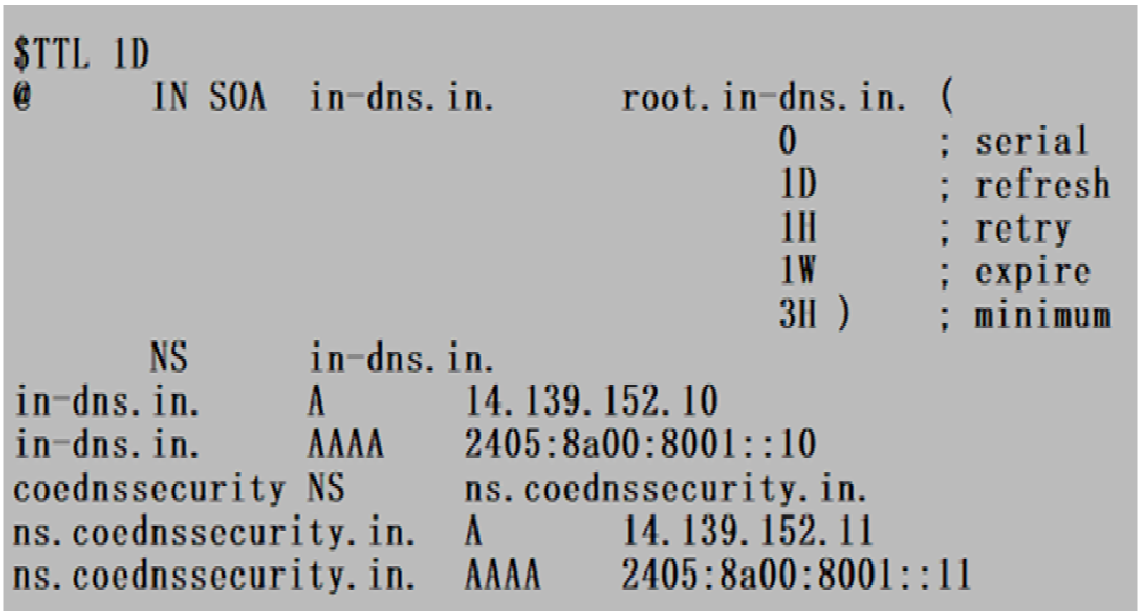
1. The "/etc/named.conf" file is modified as below mentioned:

Bind configuration file "/etc/named.conf" for "in" TLD DNS

```
options { listen-on port 53 { 14.139.152.10; };
listen-on-v6 port 53 { 2405:8a00:8001::10; };
allow-query { any; };
recursion no; };
zone "in" IN { type master;
file "in-zone"; };
```

The authoritative server for TLD "in" listens on both IPv4 and IPv6, enabling dual-stack on DNS; the queries are allowed from any host, and recursion is disabled. The server will own the "in" domain as master, and the zone file is "/var/named/in-zone".
2. Zone file configuration "/var/named/in-zone"

As shown in Figure 5.4 we added delegation to the "coednssecurity.in" zone by adding NS record "coednssecurity.in. NS ns.coednssecurity.in.". The name servers requires glue AAAA records as " ns.coednssecurity.in. AAAA 2405:8a00:8001::11" and glue A records as " ns.coednssecurity.in. A 14.139.152.11".



```
$TTL 1D
@       IN SOA  in-dns.in.       root.in-dns.in. (
                                          0       ; serial
                                          1D      ; refresh
                                          1H      ; retry
                                          1W      ; expire
                                          3H )    ; minimum
        NS      in-dns.in.
in-dns.in.      A       14.139.152.10
in-dns.in.      AAAA    2405:8a00:8001::10
coednssecurity NS       ns.coednssecurity.in.
ns.coednssecurity.in.   A       14.139.152.11
ns.coednssecurity.in.   AAAA    2405:8a00:8001::11
```

Figure 5.4 Zone File for "in" TLD DNS (/var/named/in-zone)

5.2.1.3 Installation and configuration of "coednssecurity.in" SLD server

The following are the procedures to configure the SLD "coednssecurity.in" server with Bind software:
1. The "/etc/named.conf" file is modified as below mentioned:

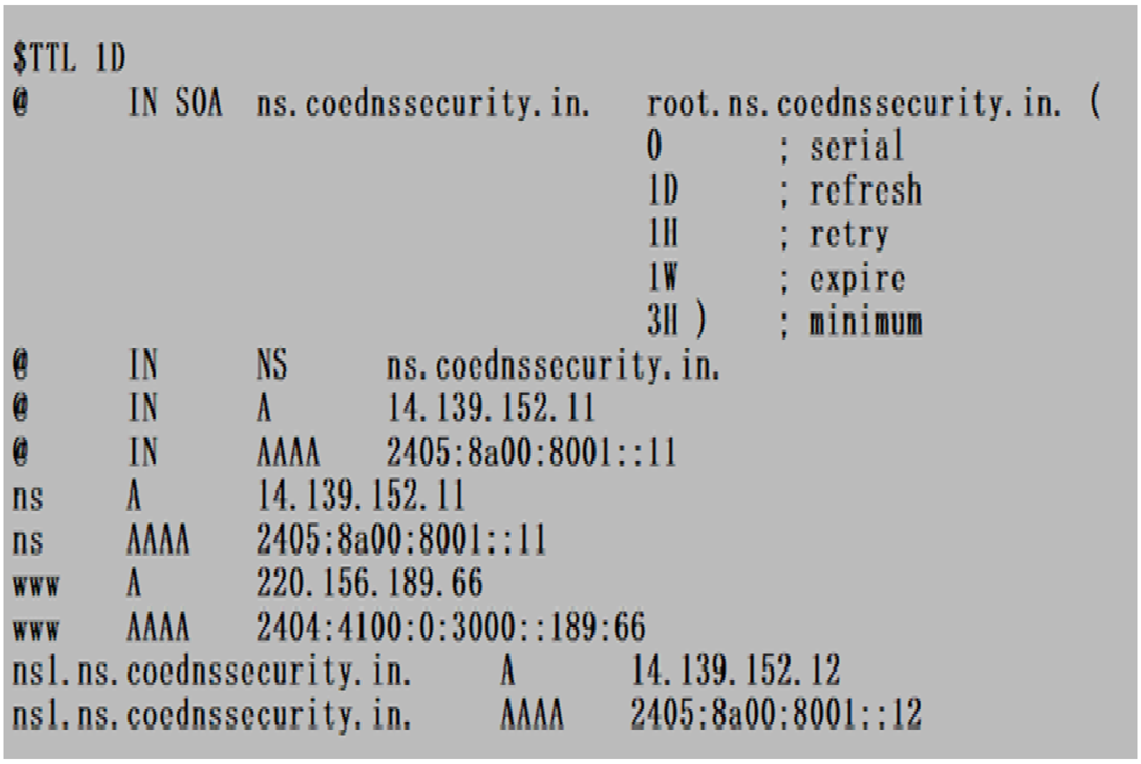
Bind configuration file "/etc/named.conf" for "coednssecurity.in" SLD DNS

```
options { listen-on port 53 { 14.139.152.11; };
listen-on-v6 port 53 { 2405:8a00:8001::11; };
allow-query { any; };
recursion no; };
zone "coednssecurity.in" IN { type master;
file "coednssecurity-zone"; };
```

The zone file for "coednssecurity.in" domain is "/var/named/coednssecurity-zone".

2. Zone file configuration "/var/named/coednssecurity-zone"

As this server is an authoritative nameserver for the "coednssecurity.in" domain, it returns the address records (both IPv4 & IPv6) for all hosts of the domain, the zone file "coednssecurity-zone" for STLD ("coednssecurity-zone.in") is shown in Figure 5.5.



```
$TTL 1D
@       IN SOA  ns.coednssecurity.in.    root.ns.coednssecurity.in. (
                                         0       ; serial
                                         1D      ; refresh
                                         1H      ; retry
                                         1W      ; expire
                                         3H )    ; minimum
@       IN      NS      ns.coednssecurity.in.
@       IN      A       14.139.152.11
@       IN      AAAA    2405:8a00:8001::11
ns      A       14.139.152.11
ns      AAAA    2405:8a00:8001::11
www     A       220.156.189.66
www     AAAA    2404:4100:0:3000::189:66
ns1.ns.coednssecurity.in.      A       14.139.152.12
ns1.ns.coednssecurity.in.      AAAA    2405:8a00:8001::12
```

Figure 5.5 Zone File for "coednssecurity.in" SLD DNS (/var/named/coednssecurity-zone)

Table 5.2 Reverse Lookup Domain Names used in the Testbed

IP Network	Reverse lookup Domain Name
IPv4 - 220.156.189	189.156.220.in-addr.arpa
IPv6 - 2404:4100:0000:3000	0.0.0.3.0.0.0.0.0.0.1.4.4.0.4.2.ip6.arpa

5.2.2 Commissioning of Reverse Lookup DNS hierarchy

Like a forward lookup tree, the reverse lookup tree is organized into a hierarchical tree of servers. Its origins may be traced back to the Address and Routing Parameter Area (arpa) top-level domain. Delegated servers "in-addr.arpa" for IPv4 and "ip6.arpa" for IPv6 are located one level below the ”arpa” TLD [97]. The reverse lookup DNS hierarchy resolves the IP address to their respective domain names; the Internet vantage point sends a DNS query asking PTR record for IP address (V4 or V6) to the recursive resolver, which queries the entire hierarchy for the result. The nodes in the reverse lookup DNS hierarchy testbed are : Root server , TLD server for "arpa" domain , SLD server for "in-addr.arpa, ip6.arpa" domains and subdomains DNS for "189.156.220.in-addr.arpa, 0.0.0.3.0.0.0.0.0.0.1.4.4.0.4.2.ip6.arpa." domains.

5.2.2.1 Setting Up Root Server for Reverse Lookup Tree

The root server setup continues following the description shown in the section-5.2.1.1. We added delegation to the "ARPA" zone by adding NS record "arpa. NS in-dns.arpa.". The name servers requires glue AAAA records as " in-dns.in. AAAA 2405:8a00:8001::10" and glue A records as " in-dns.in. A 14.139.152.10".

5.2.2.2 Setting up TLD server for "ARPA" domain

As the hierarchy needs to be established on the dual-stack network, the "arpa" domain holds information for two SLDs, "ip6.arpa" and "in-addr.arpa". The "in-addr.arpa" domain is used for IPv4 reverse DNS lookups and the ip6.arpa domain is used for IPv6 reverse lookups. In the in-addr.arpa domain, an IPv4 address is represented by a series of bytes in reverse order, expressed as decimal numbers. Dots separate the numbers, which terminate with the suffix ".in-addr.arpa.". In the "ip6.arpa" domain, an IPv6 address is represented as a name by a series of nibbles, expressed as hexadecimal digits, in

reverse order [99]. Dots separate these bits, which conclude with the suffix ". ip6.arpa.".
In our case, the examples are shown in the Table 5.2.
We configured the TLD DNS server for the "arpa" domain in the same server where the authoritative server for the "in" domain is configured, and to configure the TLD DNS server for the "arpa" domain with bind software, we followed the steps below:
1. The "/etc/named.conf" file is modified as below mentioned:

Bind configuration file "/etc/named.conf” for "ARPA" TLD DNS

```
options {listen-on port 53 { 14.139.152.10; };
listen-on-v6 port 53 { 2405:8a00:8001::10; };
allow-query { any; };
recursion no; };
zone "." IN { type hint;
file "named.ca";};
zone "in" IN { type master;
file "in-zone"; };
zone "arpa" IN { type master;
file "arpa-zone";};
```

The TLD server owns two zones, "in" and "ip6.arpa," as master with "in-zone" and "arpa-zone" zone files, respectively.
2. Zone file configuration "/var/named/arpa-zone"
As shown in Figure 5.6 we added delegation to the "in-addr.arpa" and "ip6.arpa" zones by adding NS records " in-addr.arpa. NS ns.coednssecurity.in.." and " ip6.arpa. IN NS ns.coednssecurity.in." respectively. The name servers requires glue AAAA records as " ns.coednssecurity.in. AAAA 2405:8a00:8001::11" and glue A records as " ns.coednssecurity.in. A 14.139.152.11".

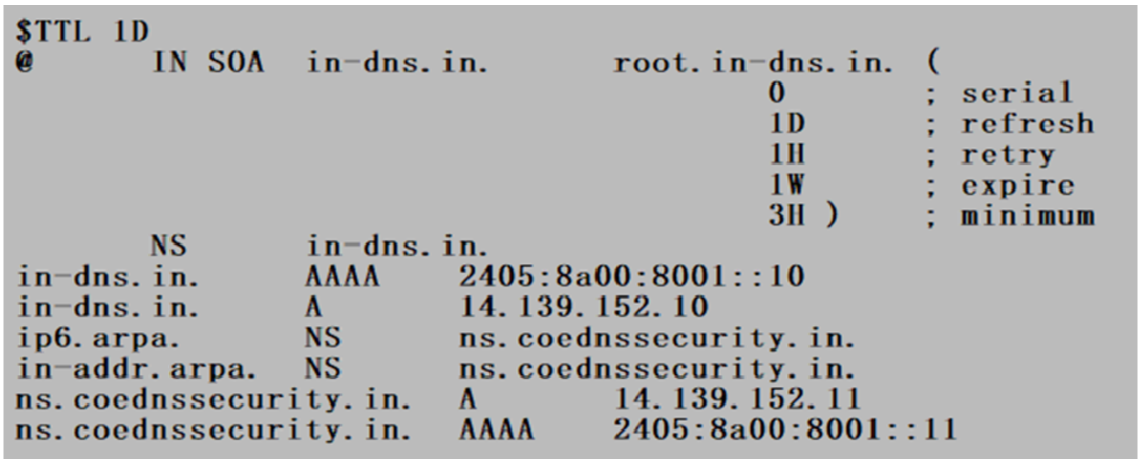

```
$TTL 1D
@       IN SOA  in-dns.in.      root.in-dns.in.  (
                                        0       ; serial
                                        1D      ; refresh
                                        1H      ; retry
                                        1W      ; expire
                                        3H )    ; minimum
        NS      in-dns.in.
in-dns.in.      AAAA    2405:8a00:8001::10
in-dns.in.      A       14.139.152.10
ip6.arpa.       NS      ns.coednssecurity.in.
in-addr.arpa.   NS      ns.coednssecurity.in.
ns.coednssecurity.in.   A       14.139.152.11
ns.coednssecurity.in.   AAAA    2405:8a00:8001::11
```

Figure 5.6 Zone File for "ARPA" TLD DNS (/var/named/arpa-zone)

5.2.2.3 Setting up SLD server for "in-addr.arpa" and "ip6.arpa" domains

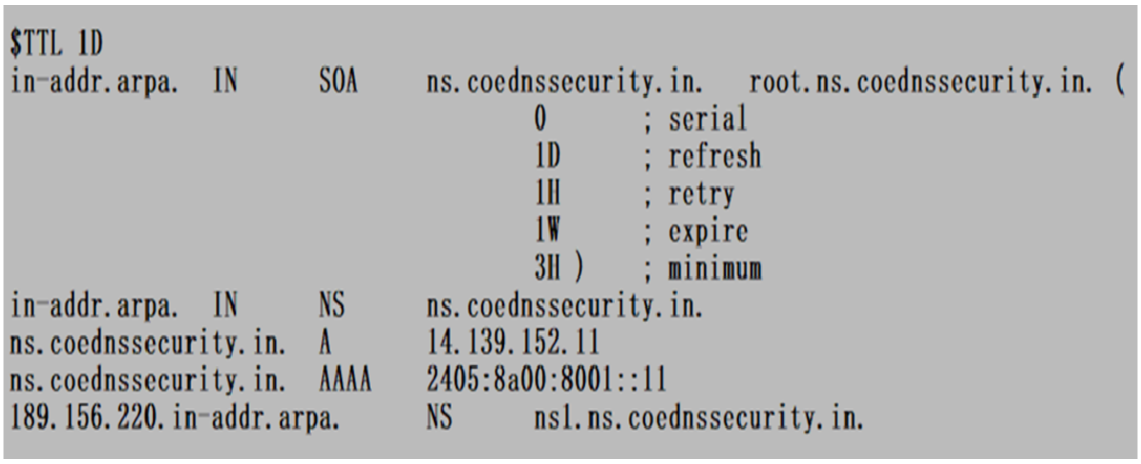
We configured the SLD server for "in-addr.arpa and ip6.arpa" domains in the same server where the authoritative DNS server for the "coednssecurity.in" domain is configured. The following procedure is followed to configure the SLD server for "in-addr.arpa" and "ip6.arpa" domains with bind software:

1. bind configuration file "/etc/named.conf" is modified as below mentioned:

Bind configuration file "/etc/named.conf" for "in-addr.arpa" and "ip6.arpa" TLD DNS

```
options { listen-on port 53 { 14.139.152.11; };
listen-on-v6 port 53 { 2405:8a00:8001::11; };
recursion no; };
zone "coednssecurity.in" IN { type master;
file "coednssecurity-zone"; };
zone "ip6.arpa" IN { type master;
file "ip6-zone";};
zone "in-addr.arpa" IN { type master;
file "in-addr-zone";};
```

Now this server becomes authoritative DNS server for three domains "coednsse-



```
$TTL 1D
in-addr.arpa.   IN      SOA     ns.coednssecurity.in.    root.ns.coednssecurity.in. (
                                        0       ; serial
                                        1D      ; refresh
                                        1H      ; retry
                                        1W      ; expire
                                        3H )    ; minimum
in-addr.arpa.   IN      NS      ns.coednssecurity.in.
ns.coednssecurity.in.   A       14.139.152.11
ns.coednssecurity.in.   AAAA    2405:8a00:8001::11
189.156.220.in-addr.arpa.       NS      ns1.ns.coednssecurity.in.
```

Figure 5.7 Zone File for "in-addr.arpa" Domain DNS (/var/named/in-addr-zone)

curity.in, in-addr.arpa and ip6.arpa" domains. The zone files for "in-addr.arpa" and "ip6.arpa" are "/var/named/in-addr-zone "and "/var/named/ip6-zone" respectively.

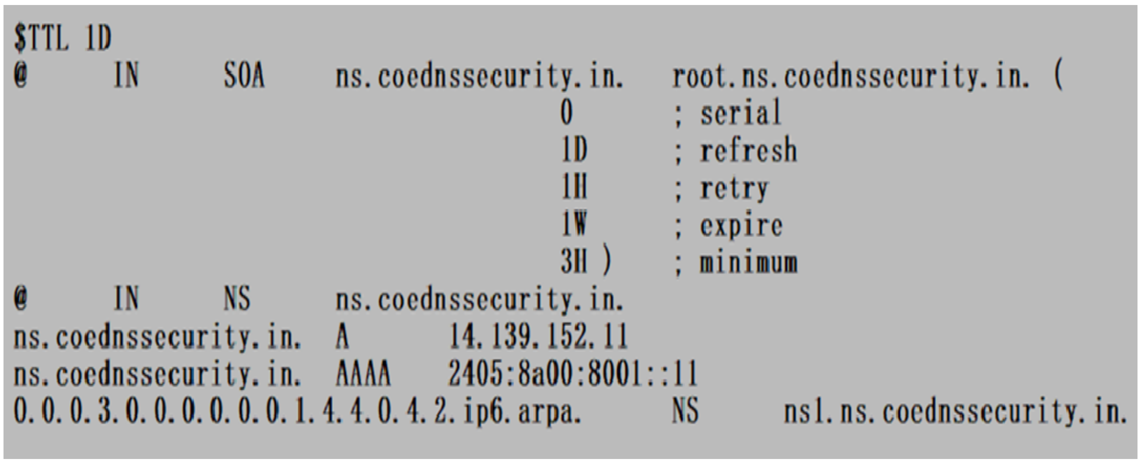
2. Zone file configuration "/var/named/in-addr.arpa"

As shown in Figure 5.7, we added delegation to the "189.156.220.in-addr.arpa." zones by adding NS records " 189.156.220.in-addr.arpa. NS ns1.ns.coednssecurity.in.".

3. Zone file configuration "/var/named/ip6.arpa"

As shown in Figure 5.8, we added delegation to the "0.0.0.3.0.0.0.0.0.0.1.4.4.0.4.2.ip6.arpa." zones by adding NS records " 0.0.0.3.0.0.0.0.0.0.1.4.4.0.4.2.ip6.arpa. NS ns1.ns.coednssecurity.in.."

The A records for ns1.ns.coednssecurity.in host needs to be added in the "coednssecu-



```
$TTL 1D
@       IN      SOA     ns.coednssecurity.in.   root.ns.coednssecurity.in. (
                                        0       ; serial
                                        1D      ; refresh
                                        1H      ; retry
                                        1W      ; expire
                                        3H )    ; minimum
@       IN      NS      ns.coednssecurity.in.
ns.coednssecurity.in.   A       14.139.152.11
ns.coednssecurity.in.   AAAA    2405:8a00:8001::11
0.0.0.3.0.0.0.0.0.0.1.4.4.0.4.2.ip6.arpa.       NS      ns1.ns.coednssecurity.in.
```

Figure 5.8 Zone File for "ip6.arpa" Domain DNS (/var/named/ip6-zone)

rity.in" SLD domain in the forward lookup tree.

5.2.2.4 Setting up Authoritative DNS Server for "189.156.220.in-addr.arpa." and "0.0.0.3.0.0.0.0.0.0.1.4.4.0.4.2.ip6.arpa " Sub-domains

The following procedures is followed to configure the authoritative server for "189.156.220.in-addr.arpa." and "0.0.0.3.0.0.0.0.0.0.1.4.4.0.4.2.ip6.arpa " sub-domains with bind software:

1. The "/etc/named.conf" file is modified as below mentioned:

Bind configuration file "/etc/named.conf" for "189.156.220.in-addr.arpa." and "0.0.0.3.0.0.0.0.0.0.1.4.4.0.4.2.ip6.arpa" Domains DNS

```
options { listen-on port 53 { 14.139.152.12; };
listen-on-v6 port 53 { 2405:8a00:8001::12; };
allow-query { any; };
recursion no;};
zone "0.0.0.3.0.0.0.0.0.0.1.4.4.0.4.2.ip6.arpa." IN { type master;
file "ipv6-reverse-zone"; };
zone "189.156.220.in-addr.arpa." IN {
type master;
file "ipv4-reverse-zone";};
```

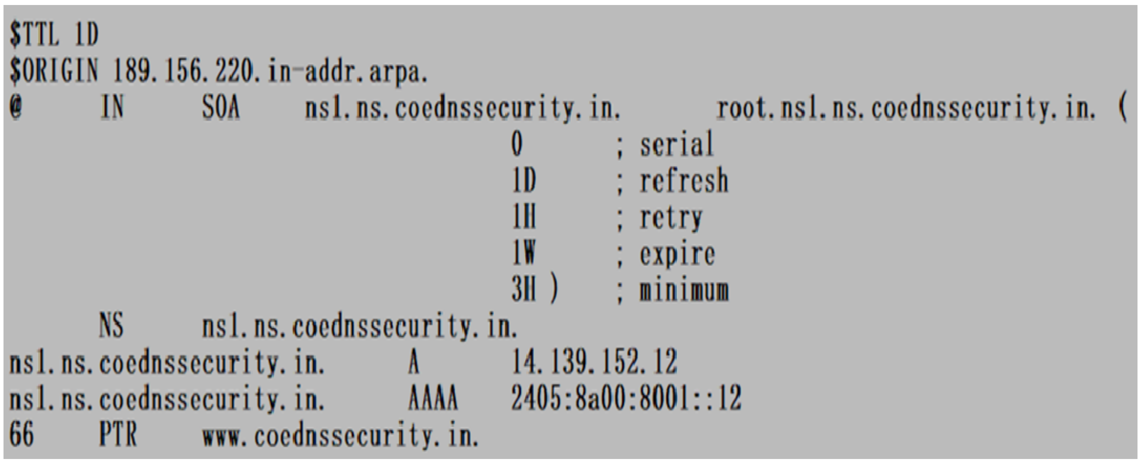
The server owns two subdomains "189.156.220.in-addr.arpa." and "0.0.0.3.0.0.0.0.0.0.1.4.4.0.4.2.ip6.arpa as master authoritative server.

2. Zone file configuration "/var/named/ipv4-reverse-zone" As shown in the Figure 5.9, we added PTR records for the IPv4 address 220.156.189.66 as "www.coednssecurity.in".

3. Zone file configuration "/var/named/ipv6-reverse-zone As shown in Figure 5.10 we added PTR records for the IPv6 address 2404:4100:0:3000::189:66 as "www.coednssecurity.in".

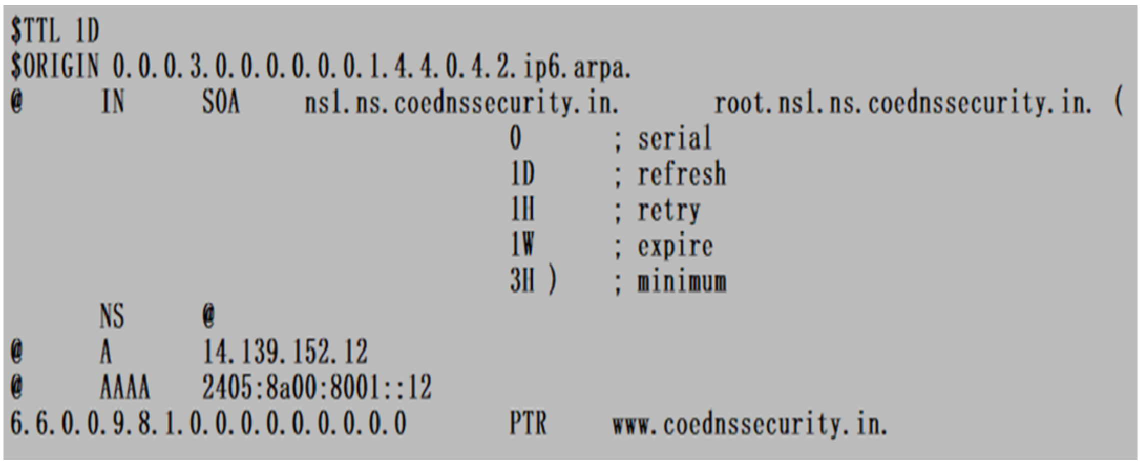
5.3 DNS Latency Measurement Methodology

When a client makes a DNS query to a recursive resolver requesting the IP address of



```
$TTL 1D
$ORIGIN 189.156.220.in-addr.arpa.
@       IN      SOA     ns1.ns.coednssecurity.in.       root.ns1.ns.coednssecurity.in. (
                                        0       ; serial
                                        1D      ; refresh
                                        1H      ; retry
                                        1W      ; expire
                                        3H )    ; minimum
        NS      ns1.ns.coednssecurity.in.
ns1.ns.coednssecurity.in.       A       14.139.152.12
ns1.ns.coednssecurity.in.       AAAA    2405:8a00:8001::12
66      PTR     www.coednssecurity.in.
```

Figure 5.9 Zone File for "189.156.220.in-addr.arpa" reverse lookup domain (/var/-named/ ipv4-reverse-zone)



```
$TTL 1D
$ORIGIN 0.0.0.3.0.0.0.0.0.0.1.4.4.0.4.2.ip6.arpa.
@       IN      SOA     ns1.ns.coednssecurity.in.       root.ns1.ns.coednssecurity.in. (
                                        0       ; serial
                                        1D      ; refresh
                                        1H      ; retry
                                        1W      ; expire
                                        3H )    ; minimum
        NS      @
@       A       14.139.152.12
@       AAAA    2405:8a00:8001::12
6.6.0.0.9.8.1.0.0.0.0.0.0.0.0.0         PTR     www.coednssecurity.in.
```

Figure 5.10 Zone file for "0.0.0.3.0.0.0.0.0.0.1.4.4.0.4.2.ip6.arpa" reverse lookup domain (/var/named/ipv6-reverse-zone)

a domain name (for example, "www.cdac.in"), the resolution process typically takes 8 query transactions, as illustrated in the Figure 5.11. The DNS query latency is the time taken to resolve the DNS query. The query latency is dependent on the hop count for each phase performed in the resolution process. The sum of the hop counts below is used to compute the overall hop count involved in the resolution of www.cdac.in:

1. The number of hops for a DNS query made from the client to the RR.
2. The number of hops for a DNS query made from RR to the root server.
3. The number of hops in a DNS reply sent from the Root server to the RR.

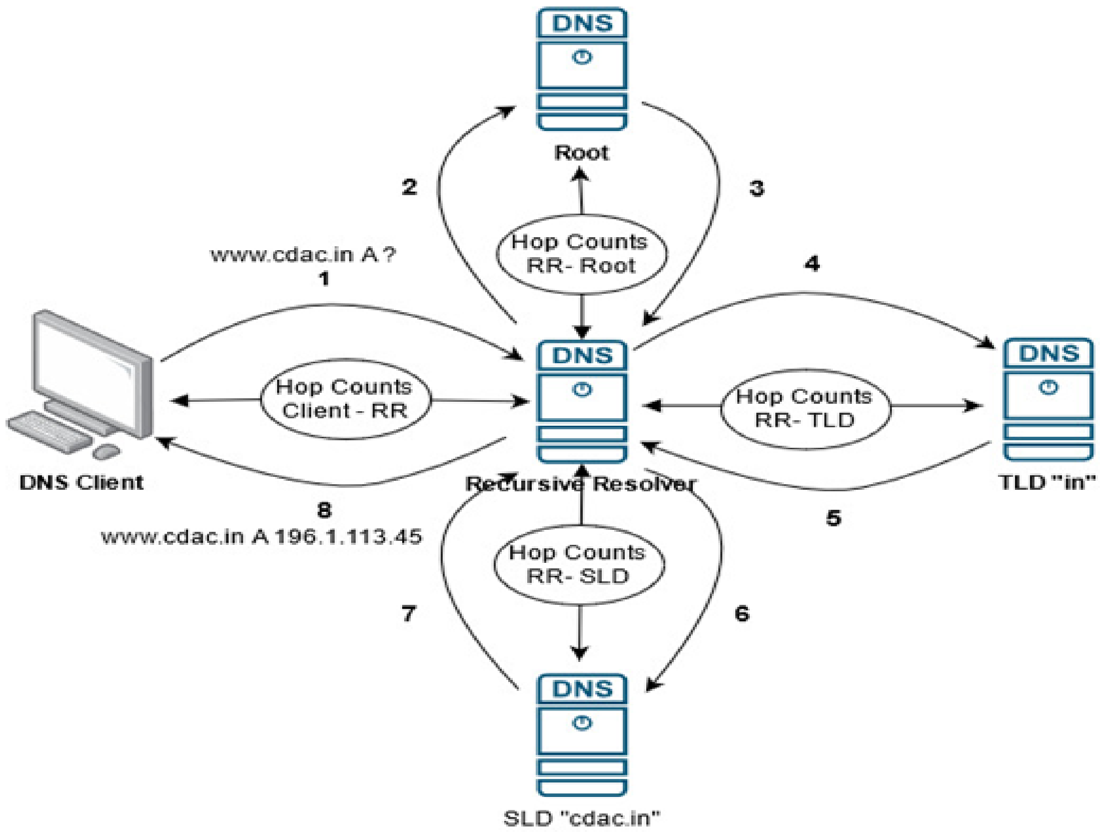


Figure 5.11 DNS Query Resolution Process involving Hop Counts

4. The number of hops for a DNS query made from RR to a TLD server.
5. The number of hops in a DNS reply sent from the TLD server to the RR.
6. The number of hops for a DNS query made from RR to an SLD server.
7. The number of hops in a DNS reply sent from the SLD server to the RR.
8. The number of hops in a DNS reply sent from the RR to the DNS client.

We aim to measure DNS latency for network layer protocols as the time taken by the recursive resolver to resolve a DNS query by searching the complete hierarchy for the domain. This is accomplished by sending a DNS query from a vantage point to RR. When the RR's DNS cache is empty, it will follow the hierarchy testbed and get the result which will be stored in its DNS cache. Then sending the identical query from the vantage point to RR again when the RR's DNS cache already has an answer for the requested query, and this time the RR replies from its cache. The latency difference between these two subsequent queries gives the DNS latency for network-layer protocols. In the Internet DNS hierarchy, there will always be a variation in the hop count for the

Table 5.3 Identified Vantage Points for Latency Test

IPv4 address	IPv6 address	Locality	Service Provider
14.139.152.3	2405:8a00:8001::120	Bangalore, India	National Knowledge Network (NKN) India
137.97.236.20	2409:4071:2107:e48c: b070:4d55:56b7:f8a6	Bangalore, India	Reliance Jio Infocomm
46.5.5.161	2a02:8070:a1b7:9c00: 8fe:40f3:8403:af14	Stuttgart Boeblingen, Germany	Liberty Global Telecommunications company
98.169.113.35	2600:8806:6100:51d: 5de3:de28:caf:8745	McLean, Virginia, USA	Cox Communications Inc
1.39.176.154	2402:3a80:cfe:b782: 51d7:299a:6dd0:fd76	Bangalore, India	Vodafone India Ltd.
86.153.4.253	2a00:23c5:b90d:700: d547:85f6:503f:dced	Farnborough , Rushmoor, England	BTnet UK Regional network

query resolution process between RR and the entire hierarchy, as it may take a different route for steps number 2 to 7. As a result, due to the variartion of hop count for two subsequent DNS quires, we cannot compare two subsequent queries for DNS latency evaluation concerning the real DNS hierarchy available on the Internet. This required setting up a DNS hierarchy testbed set up such that the hop count should always be equal from RR to each node of the hierarchy. Therefore we established a three-level DNS hierarchy comprising Root, TLD, and SLD DNS servers, and the hop count is always constant from our RR to each node in the hierarchy. For conducting a latency test, we identified few Internat vantage points across the globe, as shown in the Table 5.3. As RR and every node in the hierarchy directly connected in the same IPv4 and IPv6 network, the hop count from RR to every node is always is zero for both the network. Therefore we can precisely determine the actual DNS query latency for IPv4 and IPv6 networks, which is not achievable in the real DNS hierarchy due to the extreme variance in the number of hop counts. We created the two algorithms (9 and 10) for evaluating DNS latency for IPv4 and IP6 networks. To check DNS query latency for the IPv4 network, we disable the IPv6 interface of our RR, and we ensure the DNS cache is empty by clearing its cache. A DNS query asking for “A” record for domain “www.coednssecurity.in” is sent from a vantage point to our RR. As the cache is empty at RR, for this request, it will follow each node in the hierarchy to find the result. The result is cached, and the reply is given back to VP (A client machine or webservice on Interet from which DNS query can be sent to the hierarchy. The DNS query latency at VP for this query is recorded as “IPv4_Latency_cache_cleared”. We again send the same query from the same VP to the RR, and now this time, the reply is returned from the RR’s cache. The DNS query latency at VP for this query is recorded as “IPv4_Latency_cached”. If the hop counts (Ipv4_Hope_count_cache_cleared and IPv4_Hope_count_cached) for both subsequent queries are the same, we calculate the resulting query latency as the difference between IPv4_Latency_cache_cleared and IPv4_Latency_cached. The same method is applied to evaluate DNS query latency for IPv6 networks.

We sent the following eight types of DNS queries from each VP listed in the table by using Algorithm-9 and Algorithm-10 and recorded latency values as shown in Figure 5.12 to Figure 5.17:

1. DNS query over IPv4 network asking A record for “www.coednssecurity.in” domain, when RR cache is empty.
2. DNS query over IPv4 network asking PTR record for “2404:4100:0:3000::189:66” IP address, when RR cache is empty.
3. DNS query over IPv6 network asking A record for “www.coednssecurity.in” domain, when RR cache is empty.
4. DNS query over IPv6 network asking PTR record for “2404:4100:0:3000::189:66” IP address, when RR cache is empty.
5. DNS query over IPv4 network asking A record for “www.coednssecurity.in” domain, when RR cache has an entry for the same query.
6. DNS query over IPv4 network asking PTR record for “2404:4100:0:3000::189:66” IP address, when RR cache has an entry for the same query.
7. DNS query over IPv6 network asking A record for “www.coednssecurity.in” domain, when RR cache has an entry for the same query.
8. DNS query over IPv6 network asking PTR record for “2404:4100:0:3000::189:66”

Algorithm 9: Evaluation of DNS latency for IPv4 network

```
Input  : List of Internet Vantage Point
Output: Latency Values
1  Disable IPv6 network interface at RR
2  for every Internet Vantage Point do
3      Clear DNS cache of RR
4      Send a DNS forward lookup query from Vantage Point to RR through IPv4
         network
5      IPv4_Latency_cache_cleared = DNS query Latency measured for the query
         sent above
6      IPv4_Hope_count_cache_cleared = IPv4 Hop Count measured for the
         query sent above
7      Send a DNS forward lookup query from the same Vantage Point to RR
         through the IPv4 network
8      IPv4_Latency_cached = DNS query Latency measured for the query sent
         above
9      IPv4_Hope_count_cached = IPv4 Hop Count measured for the query sent
         above
10     if IPv4_Hope_count_cache_cleared == IPv4_Hope_count_cached then
11         IPv4_Latency = IPv4_Latency_cache_cleared - IPv4_Latency_cached
           Store IPv4_Latency for this Vantage Point.
12     end
13     else
14         goto Setp-3.
15     end
16 end
```

IP address, when RR cache has an entry for the same query.

Figure 5.18 shows the working process of DNS hierarchy testbed setup for forward lookup of “www.coednssecurity.in” domain. If the DNS cache of the RR is empty, then the resolution process will be completed in 10 query transactions starting from the Internet vantage point.

If the DNS cache of RR is empty, then the resolution process will be completed

Algorithm 10: Evaluation of DNS latency for IPv6 network

```
Input  : List of Internet Vantage Point
Output: Latency Values
1  Disable IPv4 network interface at RR
2  for every Internet Vantage Point do
3      Clear DNS cache of RR
4      Send a DNS forward lookup query from Vantage Point to RR through IPv6
         network
5      IPv6_Latency_cache_cleared = DNS query Latency measured for the query
         sent above
6      IPv6_Hope_count_cache_cleared = IPv6 Hop Count measured for the
         query sent above
7      Send a DNS forward lookup query from the same Vantage Point to RR
         through the IPv4 network
8      IPv6_Latency_cached = DNS query Latency measured for the query sent
         above
9      IPv6_Hope_count_cached = IPv4 Hop Count measured for the query sent
         above
10     if IPv6_Hope_count_cache_cleared == IPv6_Hope_count_cached then
11         IPv6_Latency = IPv6_Latency_cache_cleared - IPv6_Latency_cached
           Store IPv6_Latency for this Vantage Point.
12     end
13     else
14         goto Setp-3.
15     end
16 end
```

in 28 query transactions starting from the Internet vantage point. The same is true for reverse lookup of “220.156.199.66” IPv4 address. Figure 5.19 show for reverse lookup of “2404:4100:0:3000::189:66” IPv6 address for our testbed.

5.4 Test Results Analysis

When comparing IPv4 and IPv6 latency, it is like comparing the latency of two separate Internets. For forward and reverse lookups, the average query latency from the recursive resolver to DNS hierarchy is computed using algorithms 9 and 10 for each vantage point, as shown in Tables 5.4 and 5.5. Each node in the hierarchy and the RR are connected in the same IPv4 network and the same is true for the IPv6 network. As a result, the hop count from the recursive resolver to each node remains constant (zero hop count), giving actual DNS query latency results. We computed forward lookup la-

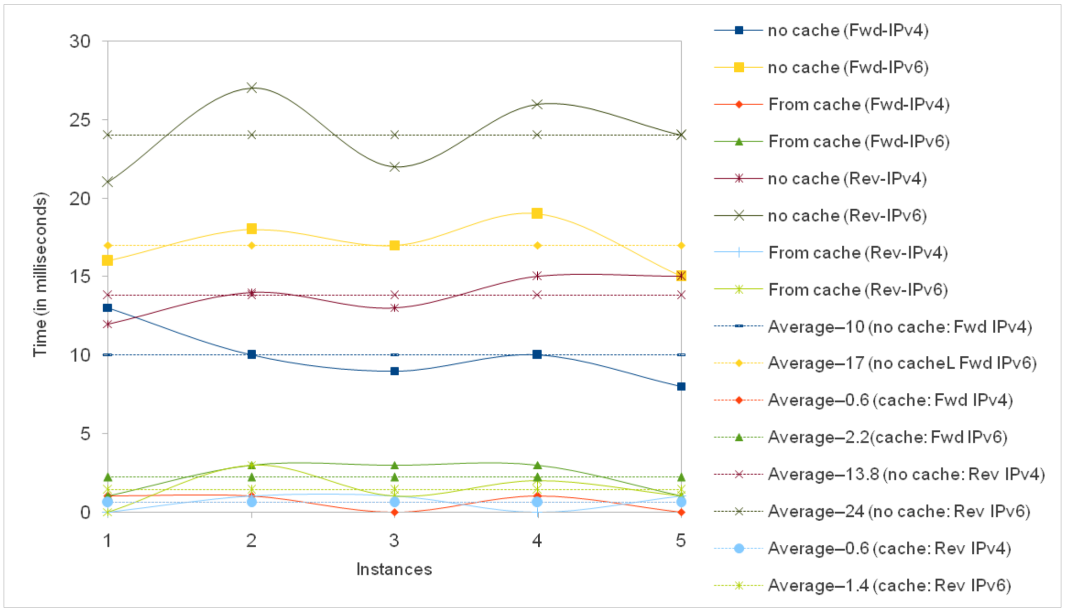


Figure 5.12 Query RTT for Forward and Reverse Lookup through " National Knowledge Network, India "

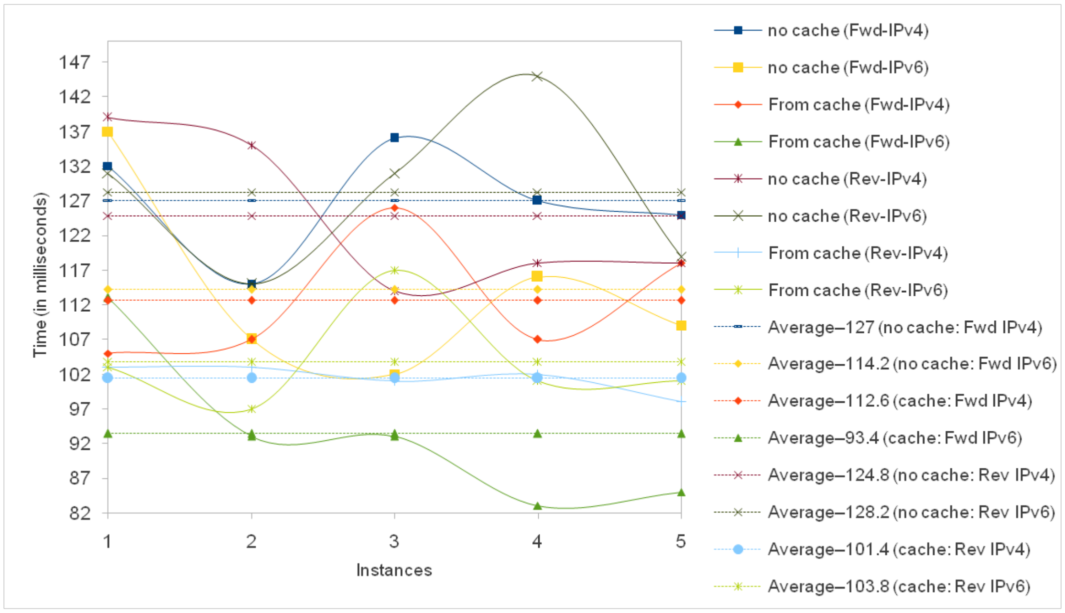


Figure 5.13 RTT for Forward and Reverse Lookup through " Reliance Jio Infocomm, India"

tency as 9.4 and 15.2 milliseconds for IPv4 and IPv6 respectively for forward lookup and 13.2, 22.6 milliseconds for reverse lookup from one vantage point of the same network of the testbed. For the testbed, the results show that IPv4 outperforms IPv6. Other vantage points, as illustrated in Tables 5.4 and 5.5, produce similar findings.

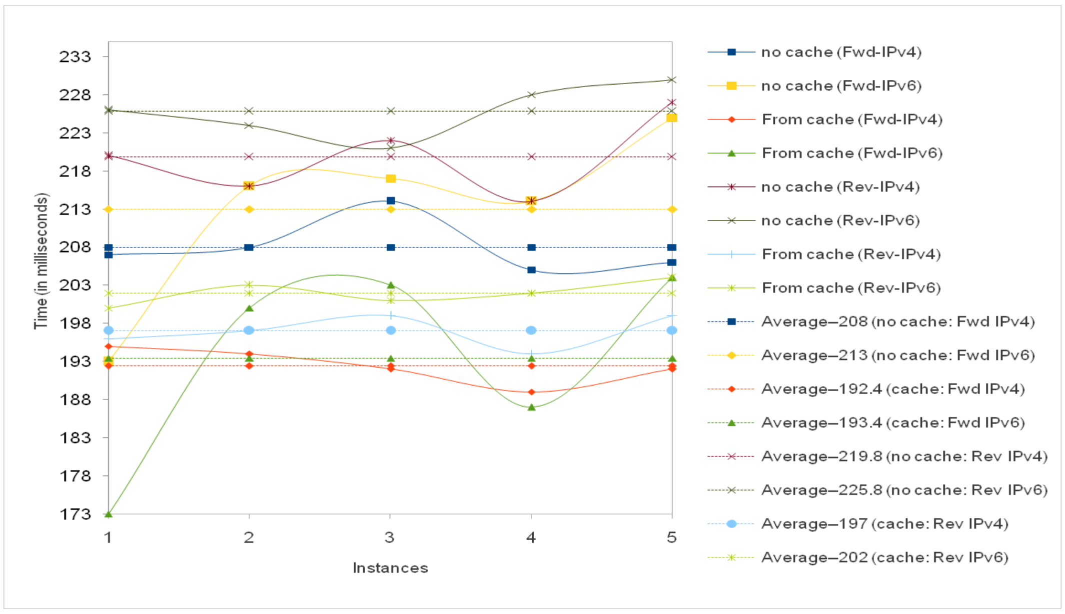


Figure 5.14 Query RTT for Forward and Reverse Lookup through "Liberty Global Telecommunications company, Germany"

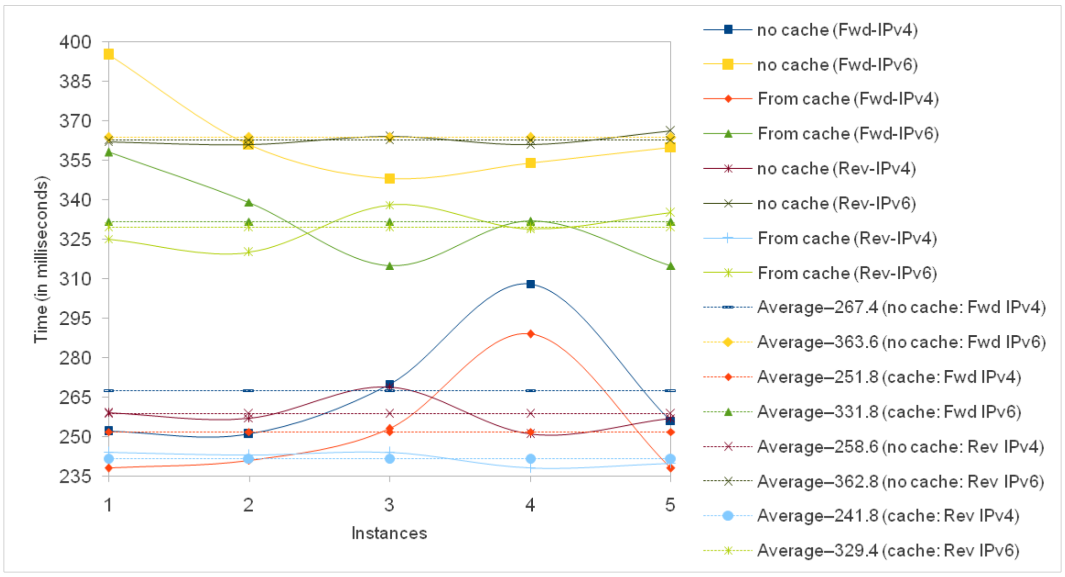


Figure 5.15 Query RTT for Forward and Reverse Lookup through "Cox Communications Inc, USA"

Since the sub-domains (189.156.220.in-addr.arpa and 0.0.0.3.0.0.0.0.0.0.1.4.4.0.4.2.ip6.arpa) are involved in the reverse lookup tree, and the total number of query transactions for a complete reverse lookup is 28 (as shown in Figures 5.19), query latency for forward lookup will always be lower than reverse lookup for both IPv4 and IPv6. According to

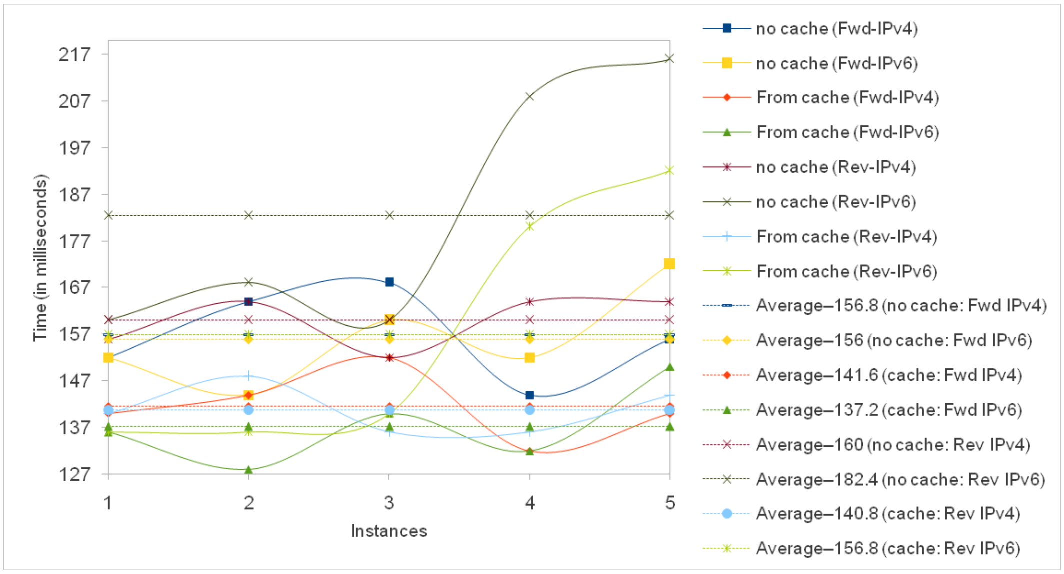


Figure 5.16 Query RTT for Forward and Reverse Lookup through " Vodafone India Ltd"

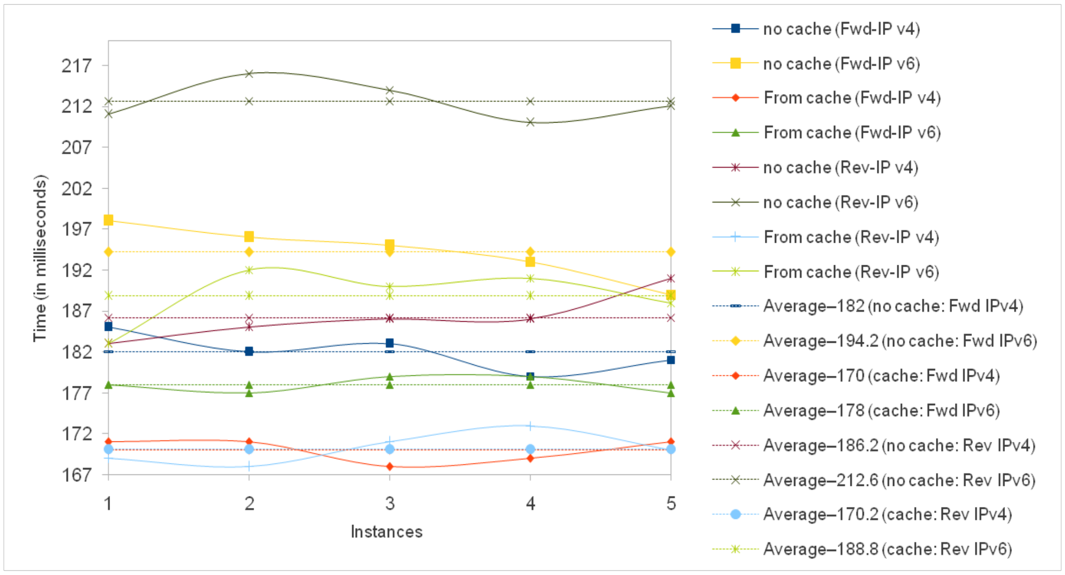
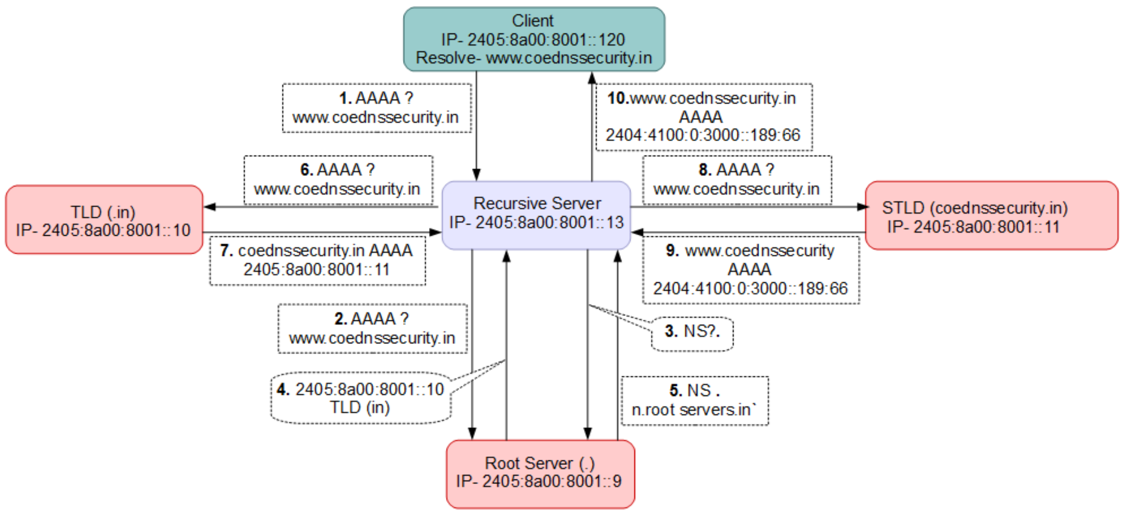


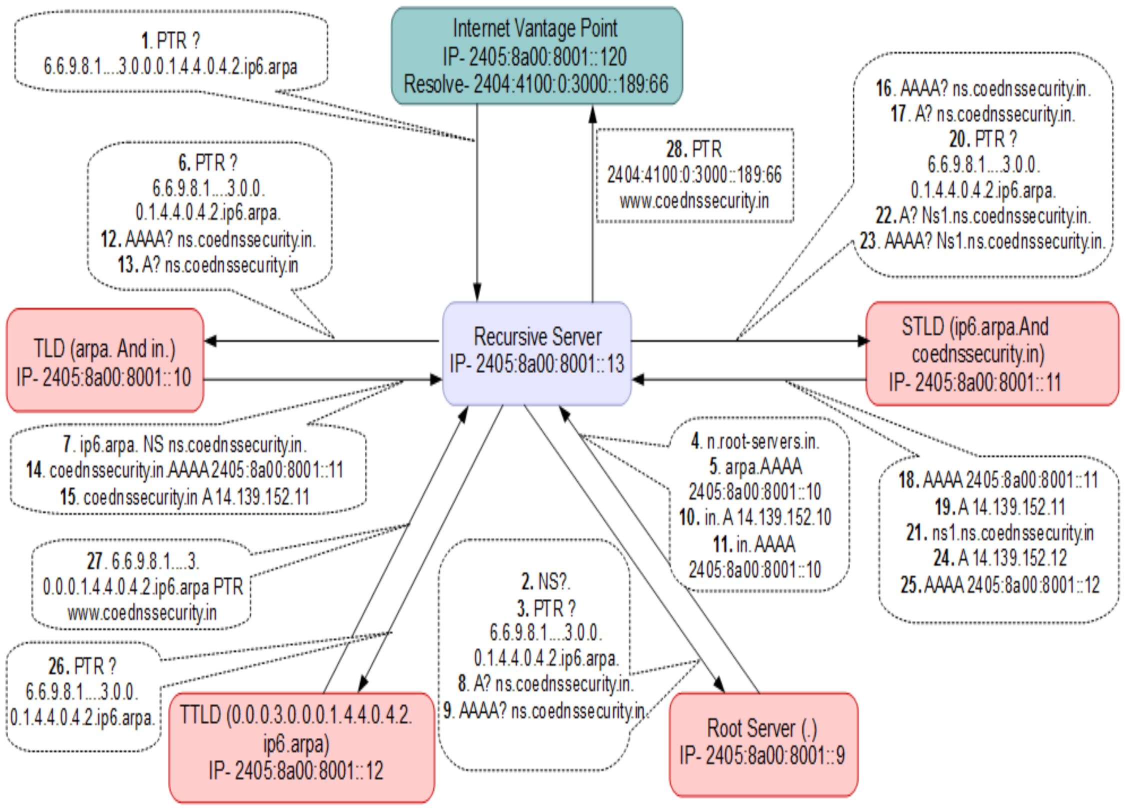
Figure 5.17 Query RTT for Forward and Reverse Lookup through " BTnet UK Regional Network"

the findings, reverse lookup latency is 35.25 percent higher than forward lookup latency for IPv4 and 25.49 percent higher for IPv6. When compared to IPv4, the findings in our experiment demonstrate that IPv6 has a modest performance loss for DNS requests. This might be because DNS query/response packet sizes are typically short, with 512 bytes or less. If DNSSEC is enabled, the results may vary since the packet size will be



Client
IP- 2405:8a00:8001::120
Resolve- www.coednssecurity.in
1. AAAA ? www.coednssecurity.in
10.www.coednssecurity.in AAAA 2404:4100:0:3000::189:66
6. AAAA ? www.coednssecurity.in
8. AAAA ? www.coednssecurity.in
TLD (.in) IP- 2405:8a00:8001::10
Recursive Server IP- 2405:8a00:8001::13
STLD (coednssecurity.in) IP- 2405:8a00:8001::11
7. coednssecurity.in AAAA 2405:8a00:8001::11
9. www.coednssecurity AAAA 2404:4100:0:3000::189:66
2. AAAA ? www.coednssecurity.in
3. NS?.
4. 2405:8a00:8001::10 TLD (in)
5. NS . n.root servers.in`
Root Server (.) IP- 2405:8a00:8001::9

Figure 5.18 The Working Process of Testbed Setup for Forward Lookup of www.coednssecurity.in'



1. PTR ? 6.6.9.8.1....3.0.0.0.1.4.4.0.4.2.ip6.arpa
Internet Vantage Point
IP- 2405:8a00:8001::120
Resolve- 2404:4100:0:3000::189:66
16. AAAA? ns.coednssecurity.in.
17. A? ns.coednssecurity.in.
20. PTR ?
6.6.9.8.1....3.0.0.
0.1.4.4.0.4.2.ip6.arpa.
22. A? Ns1.ns.coednssecurity.in.
23. AAAA? Ns1.ns.coednssecurity.in.
6. PTR ?
6.6.9.8.1....3.0.0.
0.1.4.4.0.4.2.ip6.arpa.
12. AAAA? ns.coednssecurity.in.
13. A? ns.coednssecurity.in
28. PTR
2404:4100:0:3000::189:66
www.coednssecurity.in
TLD (arpa. And in.) IP- 2405:8a00:8001::10
Recursive Server IP- 2405:8a00:8001::13
STLD (ip6.arpa.And coednssecurity.in) IP- 2405:8a00:8001::11
7. ip6.arpa. NS ns.coednssecurity.in.
14. coednssecurity.in AAAA 2405:8a00:8001::11
15. coednssecurity.in A 14.139.152.11
4. n.root-servers.in.
5. arpa.AAAA
2405:8a00:8001::10
10. in. A 14.139.152.10
11. in. AAAA
2405:8a00:8001::10
18. AAAA 2405:8a00:8001::11
19. A 14.139.152.11
21. ns1.ns.coednssecurity.in
24. A 14.139.152.12
25. AAAA 2405:8a00:8001::12
27. 6.6.9.8.1....3.
0.0.0.1.4.4.0.4.2.ip6.arpa PTR
www.coednssecurity.in
26. PTR ?
6.6.9.8.1....3.0.0.
0.1.4.4.0.4.2.ip6.arpa.
2. NS?.
3. PTR ?
6.6.9.8.1....3.0.0.
0.1.4.4.0.4.2.ip6.arpa.
8. A? ns.coednssecurity.in.
9. AAAA? ns.coednssecurity.in.
TTLD (0.0.0.3.0.0.0.1.4.4.0.4.2. ip6.arpa) IP- 2405:8a00:8001::12
Root Server (.) IP- 2405:8a00:8001::9

Figure 5.19 The Working Process of our Testbed Setup for Reverse Lookup of 2404:4100:0:3000::189:66 IP Address

Table 5.4 Average Latency for Forward Lookup Query from the Recursive Resolver to DNS Hierarchy using Algorithm 9 and 10 for each Vantage Point

ISP	IPv4 latency (L1-L2) (ms)	IPv6 latency (L3-L4)(ms)
National Knowledge Network, INDIA	9.4	15.2
Reliance Jio Infocomm, INDIA	14.4	20.8
Liberty Global Telecommunications Company, GERMANY	15.6	19.6
Cox Communications Inc, USA	15.6	31.8
Vodafone India Ltd., INDIA	15.2	18.8
BTnet UK Regional Network, ENGLAND	12	16.2
GLOBAL AVERAGE	**13.7**	**20.4**

larger.

Table 5.5 Average Latency for Reverse Lookup Query form Recursive Resolver to DNS Hierarchy using Algorithm 9 and 10 for each Vantage Point

ISP	IPv4 latency (L1-L2) (ms)	IPv6 latency (L3-L4)(ms)
National Knowledge Network, INDIA	13.2	22.6
Reliance Jio Infocomm, INDIA	23.4	24.4
Liberty Global Telecommunications Company, GERMANY	22.8	23.8
Cox Communications Inc, USA	16.6	33.4
Vodafone India Ltd., INDIA	19.2	25.6
BTnet UK Regional Network, ENGLAND	16	23.8
GLOBAL AVERAGE	**18.53**	**25.6**

5.5 Summary

We conducted a latency evaluation of DNS queries over the IPv4 and IPv6 protocol stack, by establishing a testbed emulating the DNS hierarchy of servers and a recursive resolver. The recursive resolver is probed by clients from various locations over the Internet, for resolution of a given domain name (forward lookup), and resolution of IP address (reverse lookup) in both ‘cached’ and ‘no-cache’ scenarios for computing the accurate latency caused by the different protocol stacks. The results indicate that IPv4 outperforms IPv6 in the case of DNS queries.

We listed the procedure for establishing a DNS hierarchy over dual-stack networking protocols, provisioning both forward lookup and reverse lookup, starting from ROOT, TLD (in), STLD (coednssecurity.in) for the forward lookup, and ROOT, TLD (arpa.), STLD (ip6.arpa. and in-addr.arpa.), TTLD (0.0.0.3.0.0.0.0.0.0.1.4.4.0.4.2.ip6.arpa. and 189.156.220.in-addr.arpa.) for the reverse lookup.

Chapter 6

Conclusion and Future Scope of the Work

The core of this research work deals with the safety and security of the major components of the DNS ecosystem through effective non-intrusive DNS Health Measurement and DNS Intrusion Detection. As a part of research work, we established the three-level DNS hierarchy for forward lookup and four-level hierarchy for reverse lookup on the dual-stack network and conducted latency evaluation for network-layer protocols, and figured out that DNS implementation over IPv6 has slightly more delay than IPv4.

DNS Health Measurement - A host of new parameters has been proposed to evaluate the health of the authoritative name server of any given domain along with the details of the probing algorithms for determining the value of the proposed parameters. The proposed parameters were then used for determining the health of authoritative name servers serving the top 500 domains over a period of five days and the results tabulated. This passive approach can be scaled easily across the DNS hierarchy and could be used to determine the health of the global, regional, or even a segment of the DNS system at any given instance of time. The experiment can be repeated periodically to identify divergent behavior that may lead to the abuse of authoritative name servers and prevent catastrophic failures.
The future scope of the work is extending this approach by adding more appropriate parameters (ex. Anycast support, EDNS support, etc.) to determine the health of the entire DNS hierarchy, including root name servers, TLDs name servers, and even recursive resolvers.

DNS Intrusion Detection - In this work, we propose a snort-based intrusion detection system designed particularly for detecting DNS protocol anomalies, as well as created the attack signatures for DNS tunneling, DNS amplification, and DoS/DDoS tools. Simulations on a conventional SNORT IDS deployment with inbuilt signatures revealed that the proposed solution "DID" is appropriate for attack categorization in network-based intrusion detection system for DNS-based attacks. Using various attack tools available on the Internet, we evaluated DID against a wide variety of DNS attacks (DNS amplification, tunneling, and DoS attacks). Our findings indicate a 100% attack detection rate, an extremely low false-positive alert rate of 0.01918 percent, and a 99.98 percent accuracy.
The future scope of work includes a machine learning-based approach to identifying attack patterns of DNS amplification attack tools & DNS tunneling attack tools and creating appropriate signatures dynamically.

DNS Hierarchy Testbed Setup on Dual-Stack -By constructing a testbed mimicking the DNS hierarchy of servers and a recursive resolver, we evaluated the latency of DNS requests over the IPv4 and IPv6 protocol stacks. Clients from all across the Internet probed the recursive resolver for domain name resolution (forward lookup) and IP address resolution (reverse lookup) in both cached and non-cache scenarios to compute the accurate delay caused by the various protocols stacks. In the case of DNS queries, the results show that IPv4 beats IPv6. We have also demonstrated the procedure for setting up a dual-stack enabled three-level DNS hierarchy for forward lookup and a four-level DNS hierarchy for reverse lookup. The major findings of our study show that DNS query latency in IPv6 is higher than in IPv4, and query latency in reverse lookup is always higher than forward lookup for both network-layer protocols. It also shows that when the number of DNS query repetitions for a particular query increases, the performance difference between DNS searches over IPv6 and IPv4 narrows.
The next phase of development will be to add DNSSEC to the hierarchy and perform a DNS query latency check for dual-stack, and is planed for the future scope of the work.

Appendix A

Course Work and Timeline

A.1 Course Work

S.No.	Course Code	Course Name	Credit	Grade
1.	CMA801	Computer Networks	3	AA
2.	CMA701	Data Structures And Algorithms	4	AA
3.	CS822	Topics In Computer Networks	3	AB
4.	HU800	Research Methodology (L)	2	S
5.	CS860	Information Security	3	AA
Earned Credit			15	
CGPA			9.77	

A.2 Work Timeline

Target	Dec 2017	Jan 2019	Jan 2020	Nov 2020	Sep 2021	Jan 2022
Course Work	✓					
Literature Survey	✓	✓	✓	✓	✓	✓
Research Proposal		✓				
Progress Seminar-I			✓			
Progress Seminar-II				✓		
Pre-synopsis					✓	
Thesis Submission						✓

Appendix B

List of Publications

B.1 International Journals

1. Sanjay, Balaji Rajendran, Pushparaj Shetty and Gopinath P, “Revisiting the Performance of DNS Queries on a DNS Hierarchy Testbed over Dual Stack,” accepted in The Computer Journal by Oxford Academic, UK, **2021**. (SCIE and Scopus, IF: 1.5)

2. Sanjay, Balaji Rajendran and Pushparaj Shetty, “A Quantitative Method for Measuring Health of Authoritative Name Servers,” accepted in International Journal of Information Security and Privacy (IJISP) - IGI Global, USA, **2022**. (ESCI and Scopus)

B.2 International Journals (Communicated)

1. Sanjay Adiwal, Balaji Rajendran, Pushparaj Shetty D and Sithu D Sudarsan, “DNS Intrusion Detection - DID,” communicated in The International Journal of Information Security - Springer, **2022**. (SCIE and Scopus, IF: 2.2)

B.3 International Conferences

1. Sanjay, Balaji Rajendran and Pushparaj Shetty, “Domain Name System (DNS) Security: Attacks Identification and Protection Methods,” 16th annual comprehensive security conference SAM 18(2018 Security and Management), July 30 - August 02 , **2018**, Las Vegas, USA. (EBSCO).

2. Sanjay, Balaji Rajendran and Pushparaj Shetty, “DNS Amplification & DNS Tunneling Attacks Simulation, Detection and Mitigation Approaches,” 5th International Conference on Inventive Computation Technologies (ICICT-2020), 26-28 February, **2020**, coimbatore, India. (Scopus).

References

[1] Internet Stats & Facts (2021), “List of Internet, eCommerce, Hosting, Mobile & Social Media Statistics for 2021. Websitesetup,” Available: https://websitesetup.org/news/internet-facts-stats/, [Last accessed: July, 2021].

[2] Key Internet Statistics to know in 2021, Broadband Search, Available: https://website setup.org/news/internet-facts-stats, [Last accessed: July, 2021].

[3] Hudaib, Adam Ali Zare, and E. A. Z. Hudaib. “DNS advanced attacks and analysis,” International Journal of Computer Science and Security (IJCSS), 8(2), p.63,2014.

[4] P. Mockapetris and K. J. Dunlap.“ Development of the domain name system.” Symposium proceedings on Communications architectures and protocols, 1988.

[5] Massive DDoS Attack Hit DNS Root Servers, Available: https://www.cs.cornell.edu/people/egs/beehive/rootattack.html, [Last accessed: July, 2021].

[6] Factsheet, “Root server attack on 6 February 2007”, Available: https://www.icann.org/en/system/files/files/factsheet-dns-attack-08mar07-en.pdf, [Last accessed: July, 2021].

[7] D. Massey, L. Zhang and V. Pappas, “Enhancing DNS Resilience against Denial of Service Attacks,” in 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN’07), Edinburgh, pp. 450-459, 2007.

[8] Lessons learned from Internet root server attack, Network World, Available: https://www.networkworld.com/article/2294962/update–lessons-learned-from-internet-root-server-attack.html. [Last accessed: July, 2021].

[9] Matthew Prince “The DDoS That Knocked Spamhaus Offline (And How We Mitigated It), cloud flair,” Available: https://blog.cloudflare.com/the-ddos-that-knocked-spamhaus-offline-and-ho/, [Last accessed: July, 2021].

[10] Moura, G. C., Schmidt, R. D. O., Heidemann, J., de Vries, W. B., Muller, M., Wei, L., & Hesselman, C. “Anycast vs. DDoS: Evaluating the November 2015 root DNS event,” In Proceedings of the 2016 Internet Measurement Conference, pp. 255-270, 2016.

[11] Josh Fruhlinger, “DDoS explained: How distributed denial of service attacks are evolving,” Available: https://www.csoonline.com/article/3222095/ddos-explained-how-denial-of-service-attacks-are-evolving.html. [Last accessed: Aug, 2021].

[12] Tim Greene, “How the Dyn DDoS attack unfolded,” Available: https://www.network world.com/article/3134057/how-the-dyn-ddos-attack-unfolded.html. [Last accessed: Aug, 2021].

[13] Pope, M. B., Warkentin, M., Mutchler, L. A., & Luo, X. R. “The domain name system—past, present, and future.” Communications of the Association for Information Systems, 30(1), 21, 2012.

[14] Mockapetris, P. V. “RFC1034: Domain names-concepts and facilities”, 1987.

[15] Mockapetris, P. V. “ RFC1035: Domain names - implementation and specification,” 1987.

[16] List of Top-Level Domains, IANA, Available: https://data.iana.org/TLD/tlds-alpha-by-domain.txt,[Last accessed: Aug, 2021].

[17] Weitzenboeck, E. M., “ Hybrid net: the regulatory framework of ICANN and the DNS.” International Journal of Law and Information Technology, 22(1), 49-73, 2014.

[18] DNS response message format, Available: http://www.firewall.cx/networking-topics/ protocols/domain-name-system-dns/161-protocols-dns-response.html.[Last accessed: Aug, 2021].

[19] L. Zhu, Z. Hu, J. Heidemann, D. Wessels, A. Mankin and N. Somaiya, “Connection-Oriented DNS to Improve Privacy and Security,” 2015 IEEE Symposium on Security and Privacy, pp. 171-186, 2015.

[20] Bai, X., Hu, L., Song, Z., Chen, F., & Zhao, K. " Defense against DNS man-in-the-middle spoofing." In International Conference on Web Information Systems and Mining (pp. 312-319). Springer, 2011.

[21] M. H. Jalalzai, W. B. Shahid and M. M. W. Iqbal, "DNS security challenges and best practices to deploy secure DNS with digital signatures," 2015 12th International Bhurban Conference on Applied Sciences and Technology (IBCAST), pp. 280-285, 2015.

[22] Man, K., Qian, Z., Wang, Z., Zheng, X., Huang, Y., & Duan, H. " DNS Cache Poisoning Attack Reloaded: Revolutions with Side Channels." In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pp. 1337-1350, 2020.

[23] Potasznik, M. Detecting Globally Malicious Events with Local Records: A Case Study.

[24] Kim, T. H., & Reeves, D. "A survey of domain name system vulnerabilities and attacks." Journal of Surveillance, Security and Safety, 1(1), 34-60, 2020.

[25] T. Mahjabin and Y. Xiao, "Mitigation Process for DNS Flood Attacks," 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC), pp. 1-2, 2019.

[26] T. Mahjabin, Y. Xiao, T. Li and C. L. P. Chen, "Load Distributed and Benign-Bot Mitigation Methods for IoT DNS Flood Attacks," in IEEE Internet of Things Journal, vol. 7, no. 2, pp. 986-1000, 2020.

[27] Conrad, D. (2012). "Towards Improving DNS Security, Stability, and Resiliency. Internet Society." Available: https://dev. www. isocdev. org/towards-improving-dns-security-stability-and-resiliency-0. [Last accessed: Aug, 2021].

[28] Dagon, D., Antonakakis, M., Day, K., Luo, X., Lee, C. P., & Lee, W. "Recursive DNS Architectures and Vulnerability Implications." In NDSS , 2009.

[29] Khan, I., Farrelly, W., & Curran, K. "A Demonstration of Practical DNS Attacks and their Mitigation Using DNSSEC." International Journal of Wireless Networks and Broadband Technologies (IJWNBT), 9(1), 56-78, 2020.

[30] M. Skwarek, M. Korczynski, W. Mazurczyk and A. Duda, "Characterizing Vulnerability of DNS AXFR Transfers with Global-Scale Scanning," 2019 IEEE Security and Privacy Workshops (SPW), pp. 193-198, 2019.

[31] S. Ariyapperuma and C. J. Mitchell, "Security vulnerabilities in DNS and DNSSEC," The Second International Conference on Availability, Reliability and Security (ARES'07), pp. 335-342, 2007.

[32] S. H. C. Haris, R. B. Ahmad and M. A. H. A. Ghani, "Detecting TCP SYN Flood Attack Based on Anomaly Detection," 2010 Second International Conference on Network Applications, Protocols and Services, pp. 240-244, 2010.

[33] Dhaval Kapil , DNS Security , Sep 8, 2015, Available: https://dhavalkapil.com/blogs/DNS-Security/ [Last accessed: Aug, 2021].

[34] Di Paola S., Lombardo D."Protecting against DNS Reflection Attacks with Bloom Filters." International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 1-16. Springer, Berlin, Heidelberg, 2011.

[35] Rozekrans, T., Mekking, M., & de Koning, J. "Defending against DNS reflection amplification attacks." University of Amsterdam System & Network Engineering RP1, 2013.

[36] Raman, Daan, Bjorn De Sutter, Bart Coppens, Stijn Volckaert, Koen De Bosschere, Pieter Danhieux, and Erik Van Buggenhout. "DNS tunneling for network penetration." International Conference on Information Security and Cryptology, pp. 65-77. Springer, Berlin, Heidelberg, 2012.

[37] M. Aiello, M. Mongelli and G. Papaleo, "Basic classifiers for DNS tunneling detection," 2013 IEEE Symposium on Computers and Communications (ISCC), pp. 000880-000885, 2013.

[38] Shaikh, Asadullah and Pardeshi, Bhavika and Dalvi, Faraz, “Overcoming Threats and Vulnerabilities in DNS,” Proceedings of the 3rd International Conference on Advances in Science & Technology (ICAST), 2020.

[39] GUDEKLI, U. T., & CIYLAN, B. “DNS TUNNELING EFFECT ON DNS PACKET SIZES.” 2019.

[40] Casalicchio, E., Caselli, M., Coletta, A., & Fovino, I. N. “DNS as critical infrastructure, the energy system case study.” International Journal of Critical Infrastructures 6, 9(1-2), 111-129, 2013.

[41] ICANN. “Security, Stability and Resiliency of the Domain Name System ICANN Technical Report,” 2009, Available: https://spinlock.com/wp-content/uploads/2010/02/2009-DNS-SSR-Symposium-Report.pdf [Last accessed: Aug, 2021].

[42] ICANN. “Measuring the Health of the Domain Name System.” Retrieved from Report of the 2nd Annual Global Symposium on DNS Security, Stability and Resiliency, 2010 Available: https://www.icann.org/en/system/files/files/dns-ssr-symposium-report-1-03feb10-en.pdf [Last accessed: Aug, 2021].

[43] Casalicchio, Emiliano, M. Caselli, A. Coletta, and I. Nai Fovino. “Aggregation of DNS health indicators: issues, expectations and results.” In Proc. 2012 Workshop of Securing and Trusting Internet Names, pp. 1-8. 2012.

[44] Casalicchio, Emiliano, Marco Caselli, Alessio Coletta, Salvatore Di Blasi, and Igor Nai Fovino. “Measuring name system health.” In International Conference on Critical Infrastructure Protection, pp. 155-169. Springer, Berlin, Heidelberg, 2012.

[45] E. Casalicchio, M. Caselli and A. Coletta, “Measuring the global domain name system,” in IEEE Network, vol. 27, no. 1, pp. 25-31, 2013.

[46] Cy, Tejaswini Yadav, Balaji Rajendran, and P. rajani. “An Approach for Determining the Health of the DNS.”, International Journal of Computer Science and Mobile Computing,Vol. 3, Issue. 9, pg.442 – 449, 2014.

[47] Yamini, C., Balaji, R., & Papanna, M. N., “DNS Health Visualizatio”, International Journal of Computer Science and Mobile Computing,V ol. 3, Issue. 9, pg.202 – 211, 2014.

[48] Keyu Lu, Zhengmin Li, Zhaoxin Zhang and Jiantao Shi, “DNS recursive server health evaluation model,” 18th Asia-Pacific Network Operations and Management Symposium (APNOMS), pp. 1-4, 2016.

[49] C. Deccio, “DNS Diagnostics through the Eye of the Beholder,” International Conference on Computing, Networking and Communications (ICNC), pp. 753-758, 2019.

[50] R. Munadi, E. Firdaus, T. Y. Arif and F. Y. Zulkifli, “An evaluation of DNS server health of public universities in Sumatera Island,” 15th International Conference on Quality in Research (QiR) : International Symposium on Electrical and Computer Engineering, pp. 13-17, 2017.

[51] Jian Jin, Zhiwei Yan, Jong-Hyouk Lee, Guanggang Geng, “Health Evaluation of a Domain Name System Based on the Analytic Hierarchy Process,” Journal of Internet Technology, vol. 19, no. 1 , pp. 027-034, 2018.

[52] Jaeyeon Jung, E. Sit, H. Balakrishnan and R. Morris, “DNS performance and the effectiveness of caching,” IEEE/ACM Transactions on Networking, vol. 10, no. 5, pp. 589-603, 2002.

[53] Richard Liston, Sridhar Srinivasan, and Ellen Zegura, “ Diversity in DNS performance measures,” Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment (IMW ’02). Association for Computing Machinery, New York, NY, USA, 19–31, 2002.

[54] Y. Wu, J. Tuononen and M. Latvala, “Performance Analysis of DNS with TTL Value 0 as Location Repository in Mobile Internet,” IEEE Wireless Communications and Networking Conference, pp. 3250-3255, 2007.

[55] Domain DNS Health Checker Tool, Available: https://dnschecker.org/domain-health-checker.php [Last accessed: Sept, 2021].

[56] Paranoid about your DNS. Monitor, validate and verify your DNS configurations, Available: https://dnsspy.io/ [Last accessed: Sept, 2021].

[57] Domain Health Report, Available: https://mxtoolbox.com/domain/ [Last accessed: Sept, 2021].

[58] DNS health check : find bugs on your domain, Available: https://www.dnsqueries .com/en/domain_check.php [Last accessed: Sept, 2021].

[59] DNS Monitoring Tool for Server Performance and Health Checks, Available: https://www.solarwinds.com/ server-application-monitor/use-cases/dns-monitoring [Last accessed: Sept, 2021].

[60] IntoDNS checks the health and configuration and provides DNS report and mail servers report, Available: https://intodns.com / [Last accessed: Sept, 2021].

[61] Drozdova, A. "SECURING A DNS SERVER WITH SNORT IDS: Severen-Telecom case", 2015.

[62] Hock, F., & Kortiš, P. "Design, implementation and monitoring of the firewall system for a DNS server protection." International Conference on Emerging eLearning Technologies and Applications (ICETA) (pp. 91-96). IEEE, 2016.

[63] Kim, B. H., & Park, Y. G. "Design and analysis of client control system using DNS control firewall." International Journal of Smart Home, 7(5), 135-144, 2013.

[64] Afonso J, Veiga P. "Improving DNS Security Using Active Firewalling with Network Probes." International Journal of Distributed Sensor Networks. May 2012.

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

[66] Satam, P., H. Alipour, Youssif B. Al-Nashif and S. Hariri. "Anomaly Behavior Analysis of DNS Protocol." J. Internet Serv. Inf. Secur. 5, 85-97, 2015.

[67] Cheung, Steven, and Karl N. Levitt. "A formal-specification based approach for protecting the domain name system." Proceeding International Conference on Dependable Systems and Networks, IEEE, 2000.

[68] Rastegari, Samaneh & Saripan, M Iqbal & Rasid, Mohd. "Detection of Denial of Service Attacks against Domain Name System Using Neural Networks." International Journal of Computer Science Issues. 6, 2009.

[69] Robert J. Shimonski, Wally Eaton, Umer Khan, Yuri Gordienko, "Chapter 11 - Detecting and Performing Security Breaches with Sniffer Pro", Sniffer Pro Network Optimization and Troubleshooting Handbook, Syngress, Pages 513-565, 2002.

[70] Liang, J., Jiang, J., Duan, H., Li, K., & Wu, J. "Measuring query latency of top level DNS servers." International Conference on Passive and Active Network Measurement, pp. 145-154, 2013.

[71] Sammour, M., Hussin, B., Othman, M. F. I., Doheir, M., AlShaikhdeeb, B., & Talib, M. S. "DNS Tunneling: a Review on Features." International Journal of Engineering and Technology, 7(3.20), 1-5, 2018.

[72] Efficient IP, "DNS Security, Secure Your DNS, Secure Your Network," Available: https:// www.efficientip.com/solutions/dns-security/ [Last accessed: Sept, 2021].

[73] AKAMAI, "The State of the Internet," Available: https://www.akamai.com/uk/en/ multimedia/documents/state-of-the-internet/dnssec-amplification-ddos- security-bulletin.pdf [Last accessed: Sept, 2021].

[74] Zhao, G., Xu, K., Xu, L., & Wu, B. "Detecting APT malware infections based on malicious DNS and traffic analysis." IEEE access, 3, 1132-1142, 2015.

[75] Farnham, G., & Atlasis, A. " Detecting DNS tunneling." SANS Institute InfoSec Reading Room, 9, 1-32, 2013.

[76] M. Bretelle, “Encoding DNS-over-TLS (DoT) Subject Public Key Info (SPKI) in Name Server name.” Available: https://tools.ietf.org/id/draft-bretelle-dprive-dot-spki-in-ns-name-00.html, [Last accessed: Oct, 2021]

[77] Verisign, The Domain Name Industry Brief, Available: https://www.verisign.com/en_US/domain-names/dnib/index.xhtml, [Last accessed: Oct, 2021]

[78] C. Deccio, J. Sedayao, K. Kant and P. Mohapatra, “Measuring Availability in the Domain Name System,” 2010 Proceedings IEEE INFOCOM, pp. 1-5, 2010.

[79] Chandramouli, R., & Rose, S. “Secure domain name system (DNS) deployment guide.” NIST Special Publication, 800, 81-2, 2006.

[80] Narayan, S., Peng S., and Fan, N. “Network performance evaluation of Internet Protocols IPv4 and IPv6 on operating systems.” Proceedings of 2009 IFIP International Conference on Wireless and Optical Communications Networks, Cairo, Egypt, 28-30 April, pp. 1-5, IEEE, 2009.

[81] Fredrik Ljunggren, Jakob Schlyter, Kirei AB, DNS server fingerprinting tool - "fpdns" , Available: https://github.com/kirei/fpdns, [Last accessed: Oct, 2021]

[82] Al-Dalky, R. and Rabinovich, M. “Revisiting Comparative Performance of DNS Resolvers in the IPv6 and ECS Era.” arXiv:2007.00651 [cs.NI],2020.

[83] Moz’s list of the most popular 500 websites on the internet. Available: https://moz .com/top500, [Last accessed: Nov, 2021]

[84] Yue Wang, Anmin Zhou, Shan Liao, Rongfeng Zheng, Rong Hu, Lei Zhang, “A comprehensive survey on DNS tunnel detection,” Computer Networks,Volume 197, 2021.

[85] Bjorn Andersson and Erik Ekman, IODINE, Available: https://code.kryo.se/iodine, [Last accessed: Nov, 2021]

[86] Gudekli, U. T., & Ciylan, B. “dns tunneling effect on dns packet sizes.”, 2019.

[87] ThunderDNS, Available: https://blog.fbkcs.ru/traffic-at-the-end-of-the-tunnel-or-dns-in-pentest/, [Last accessed: Nov, 2021]

[88] Rossow, C.“Amplification Hell: Revisiting Network Protocols for DDoS Abuse.” In NDSS, 2014.

[89] Ethanwilloner, DNS-Amplification-Attack, Available: https://github.com/ethan willoner/DNS-Amplification-Attack, [Last accessed: Nov, 2021]

[90] Offensive python/ Saddam - DDoS Tool for Amplification, Available: https://github.com/ OffensivePython/Saddam, [Last accessed: Nov, 2021]

[91] Prolexic, DNS Flooder v1.1, Available: https://github.com/plxsertr/dnsreflect, [Last accessed: Nov, 2021]

[92] DNS dist ributed reflection Denial of Service, noptrix, Available: ht tp://www.nopt rix.net, [Last accessed: Nov, 2021]

[93] Google Code, DNS-Flood, Available: https://code.google.com/archive/p/dns-flood/, [Last accessed: Nov, 2021]

[94] Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity, 2(1), 1-22.

[95] Rose, K., Eldridge, S., & Chapin, L. “The internet of things: An overview.” The internet society (ISOC), 80, 1-50, 2015.

[96] Hanumanthappa, J., & Manjaiah, D. H. “A Study on Comparison and Contrast between IPv6 and IPv4 Feature Sets.” In International Conference on Computer Network and Security (ICCNS), 2008.

[97] Ploessel, M. “Attacking and Defending DNS.” In Special Ops (pp. 393-452). Syngress, 2003.

[98] Nikkel, B. J. “An introduction to investigating IPv6 networks.” Digital Investigation, 4(2), 59-67, 2007.

[99] Hu, Q., & Brownlee, N. “IPv6 host address usage survey.” International Journal of Future Computer and Communication, 3(5), 341, 2014.

[100] Liang Zhu and John Heidemann. “LDplayer: DNS Experimentation at Scale.” In Proceedings of the Internet Measurement Conference 2018 (IMC ’18). Association for Computing Machinery, New York, NY, USA, 119–132, 2018.

[101] L. Song, Ed., D. Liu, P. Vixie, A. Kato, S. Kerr, Yeti DNS Testbed, RFC8483, Available: https://datatracker.ietf.org/doc/html/rfc8483, [Last accessed: Nov, 2021]

[102] Park, S., Jeong, J. and Hong, C.S. “DNS Configuration in IPv6: Approaches, Analysis, and Deployment Scenarios.” IEEE Internet Computing, 17, 48-56, 2013.

[103] Fuliang, L., Xingwei, W., Tian, P. and Jiahai, Y. “A Case Study of IPv6 Network Performance: Packet Delay,Loss, and Reordering.” Hindawi Mathematical Problems in Engineering, 2017.

[104] Soorty, B., & Sarkar, N. I. “Evaluating IPv6 in peer-to-peer Gigabit Ethernet for UDP using modern operating systems.” In 2012 IEEE Symposium on Computers and Communications (ISCC) (pp. 000534-000536). IEEE, 2012.

[105] Xiaoming, Z., Jacobsson, M., Uijterwaal, H., and Mieghem, P. “IPv6 delay and loss performance evolution.” Int. J. Commun. Syst., 21, 643-663, 2008.

[106] Savita, S., Purohit, G.N., and Naveen, H. “Performance Analysis of IPv4 v/s IPv6 in Virtual Environment using UBUNTU.” Proceedings of International Conference on Computer Communication and Networks CSI-COMNET-2011, Udaipur, India, 4-6 December, pp. 86-91, IJCA,2011.

[107] D. Barr, “RFC1912: Common DNS Operational and Configuration Errors”, 1996.