

# Cache poisoning attack over DNS

Vintilă Diana-Georgiana, Grosu Ștefan

January 18, 2023

## Abstract

Domain Name System (DNS) is a critical internet protocol that provides a way to resolve domain names to their corresponding IP addresses. DNS is known to be vulnerable to a widespread attack called DNS poisoning. Our article aims to study the Domain Name System and the mechanism of DNS cache poisoning. Our paper will present countermeasures used against this type of attack and why is important that future research should be oriented towards introducing more trustworthy countermeasures against DNS cache poisoning.

## 1 Introduction

### 1.1 What is DNS ?

People access information online through domain names, like google.com or facebook.com. Web browsers interact through Internet Protocol (IP) addresses. DNS translates domain names to IP addresses so browsers can load Internet resources.

An IP address is associated with each device on the Internet and that address is necessary to identify the appropriate Internet device. When a user wants to load a webpage, a translation must occur between user typing (example.com) and the machine-friendly address, to locate the example.com webpage.

### 1.2 How DNS works?

DNS is considered to be a distributed “database” to resolve domain names structured by a tree data structure and each node has a label and zero or more resource records. Domain names are always assigned by the Internet Assigned Numbers Authority (IANA). A domain represents the entire set of nodes that are under a specific top level note from that domain.

DNS servers use both TCP and UDP protocols. TCP is usually used for zone transport, for example for discussion between master and slaves. In other configurations UDP protocol will be used. If the response to the client is larger than a limit of 5012 bytes, then the response will be sent not by UDP but by TCP connection.

Each domain can have multiple zones, each managed by only one DNS name server. Each DNS server has a zone file containing all the resources managed by that server. The indicator “NS” (Name server) specifies all the servers present in that zone. In case there is only one master server with several slave servers, the DNS master server will be called “SOA” (Start of authority). The servers should have the same knowledge of the zone. The master will broadcast the serial number from time to time to update the slave servers’ information.

### 1.3 DNS architecture?

Usually, there are four DNS servers involved in loading a webpage: a DNS recursor (also called a recursive resolver), root nameserver, top-level domain (TLD) nameserver and authoritative nameserver. The DNS recursor is a server designed to receive queries from client machines through applications such as web browsers while the root nameserver is the first step in translating (resolving) human readable host names into IP addresses, serving as a reference to other more specific locations.

The top-level domain server is the next step in the search for a specific IP address and it hosts the last portion of a hostname (e.g. in example.com, the TLD server is “com”), being followed by the final component, the authoritative nameserver which is the last in the nameserver query.

If the authoritative name server has access to the requested record, it will return the IP address for the requested hostname back to the DNS Recursor (the user input) that made the initial request.

## 1.4 What is DNS cache poisoning?

DNS spoofing is the act of entering false information into a DNS cache, so that DNS queries return an incorrect response and users are directed to the wrong websites.

DNS resolver cache saves responses to IP address queries for a certain amount of time. In this way, the resolver can respond to future queries much more quickly, without needing to communicate with the many servers involved in the typical DNS resolution process. Because there is typically no way for DNS resolvers to verify the data in their caches, incorrect DNS information remains in the cache until the time to live (TTL) expires, or until it is removed manually.

Attackers can poison DNS caches by impersonating DNS nameservers, making a request to a DNS resolver, and then forging the reply when the DNS resolver queries a nameserver. This is possible because DNS servers use UDP instead of TCP, and because currently there is no verification for DNS information.

### 1.4.1 Cache poisoning vs phishing?

Though DNS cache poisoning has similar end results to phishing: getting a user to believe a bad site is legitimate, it’s not the same thing. With phishing, a fake URL is used to reference the malicious server, but it’s disguised to look like the real hostname (usually with a bit of clever HTML or CSS). An user can usually detect this misdirection by examining the URLs or hostnames carefully.

Instead of using TCP, which requires both communicating parties to perform a ‘handshake’ to initiate communication, DNS requests and responses use UDP. With UDP, there is no guarantee that a connection is open or that the recipient is ready to receive. UDP is vulnerable to forging for this reason – an attacker can send a message via UDP and pretend it is a response from a legitimate server by forging the header data.

DNS queries are not encrypted. Even if users use a DNS resolver that does not track their activities, DNS queries travel over the Internet in plaintext. This means anyone who intercepts the query can see which websites the user is visiting.

Attackers also have to either know or guess a number of factors to carry out DNS spoofing attacks:

- which DNS queries are not cached by the targeted DNS resolver so that the resolver will query the authoritative nameserver;
- what port the DNS resolver is using – they used to use the same port for every query, but now they use a different, random port each time;
- the request ID number;
- which authoritative nameserver the query will go to.

Attackers could also gain access to the DNS resolver using malicious party operates, hacks, or gains physical access to it.

## 1.5 What are DNS inherent vulnerabilities?

Users and hosts trust the host-address mapping provided by DNS. Also, DNS resolvers trust responses received after sending out queries. Responses can include DNS information unrelated to the query.

## 2 Related work attacks description

### 2.1 First presentation of this attack

In 2008, Dan Kaminsky, a security researcher, presented a DNS vulnerability that allowed attackers to send users to malicious sites and hijack email at security conference Black Hat. The exploit would allow attackers to impersonate any legitimate website and steal data.

### 2.2 Types of attacks

There are two types of DNS cache poisoning attacks: forging and defragmentation attacks.

#### 2.2.1 Forging Attacks

Forging attacks suppose guessing the metadata such as: TXID, src port. We will discuss some examples of these class of attacks: Birthday Attack and Kaminsky Attack.

##### Birthday attack

- Force the resolver to send many identical queries, with different IDs, at the same time.
- Increase the probability of making a correct guess.

##### The Kaminsky Attack

- This is a sort of evolution of the basic cache poisoning attack, in which the victim is an entire domain instead of a single host.
- The aim of the attack is to feed a victim recursive DNS server with a forged response packet to spoof an NS record.

**Mitigation:** based on randomization (RFC 5452).

#### 2.2.2 Defragmentation Attacks

These types of attacks are targeting DNS forwarders.

An example of this type of attack is **SAD DNS Attack with side-channels**.

The attack exploits the fact that the second fragment of a fragmented DNS response packet does not contain DNS or UDP headers or question section, so it can bypass randomization-based defenses against forging attacks.

**Mitigation:** based on verifying the oversized DNS response or 0x20 encoding on DNS records.

### 2.3 Poisoning the cache

DNS only accepts responses to pending queries. The next question arises: How does a nameserver know that any response packet is "expected"? The answer is simple: the response is received on the same UDP port that it was sent from, the Question section and Query ID the pending query and the Authority and Additional sections represent names that are within the same domain as the question ("bailiwick checking").

If all of these conditions are satisfied, a nameserver will accept a packet as an authentic response to a query, and use the results found inside, including caching answers, as well as valid authority and additional data, found there too.

##### Mitigation

One possible mitigation would be to randomize the Query ID. If the nameserver chooses random Query IDs, then the attacker has the full 16-bit pool to choose from, giving 64k values to guessing the narrow window of time while the victim is going through the resolution steps.

Randomizing IPID values. Random IPID values makes any defragmentation-based attacks (including ours) much more difficult, as they require the prediction of future IPID values. Interestingly, as we

have tested and described earlier in Section 4.3, major operating systems such as Windows and Linux do not exhibit such a random IPID behavior.

Yet in our measurement, we do find Google and Versign’s resolvers appear to have such behaviors. We suspect that it is either because they have used uncommon/customized operating systems and network middleboxes (that rewrite the IPIDs), or that there are actually still multiple hosts sitting behind the same egress IP address (e.g., through NAT).

In any event, random IPID values are not impossible to guess, especially given that the attacker can place 64 guessed values (out of 64K possible values). Furthermore, if the attack is repeated multiple times, the likelihood of success will increase as well. As a result, it is not a bullet-proof mitigation.

## 2.4 Security risks of DNS forwarders

A DNS forwarder does not perform recursive DNS lookup themselves, but simply forwards DNS requests to an upstream resolver. In order to mitigate the security risks of DNS cache poisoning and denial of service attacks, DNS forwarders are widely implemented in network products related to DNS protocol, such as home routers, as it is not directly exposed to Internet attackers. It is also recommended by some DNS experts, e.g., Kaminsky.

Unfortunately, many DNS forwarders themselves are not patched and are vulnerable to DNS cache poisoning attacks. In some cases, DNS forwarders fail to validate the DNS responses, such as the DNS transaction ID, source IP address and destination port number. A measurement study shows that at least 8.6% open DNS resolvers in the wild are vulnerable to DNS cache poisoning attacks. Therefore, in spite of the availability of DNSSEC, DNS record injection vulnerabilities are still fairly common among DNS forwarders until now.

Compared to previous works, in this paper, we further present a type of cache poisoning attack targeting DNS forwarders. The methods can circumvent traditional defenses against cache poisoning attacks. Combined with previous attacks, our work further demonstrates that DNS forwarders can be a soft spot in the infrastructure

## 3 Effects attack mitigation

Security mechanisms developed for DNS servers are UDP Source Port Randomization (UDP SPR) and DNS Security Extensions (DNSSEC).

UDP Source Port Randomization is setting the UDP source port randomly, so an attacker would have to guess both the transaction ID and the source port in a short time window - which is usually not feasible (since they would need to make  $2^{32}$  combinations).

DNS was built initially for a much smaller Internet and based on a principle of trust (much like BGP). A more secure DNS protocol called DNSSEC aims to solve some of these problems, but it has not been widely adopted yet. DNSSEC is an abbreviation for Domain Name System Security Extensions, and it is a means of verifying DNS data integrity and origin. DNSSEC is designed to create a unique cryptographic signature and store it alongside other DNS records. This is done on all levels of the DNS Resolution process.

The most effective way to mitigate DNS cache poisoning consequences for your own web assets is to enforce HTTPS connections by using HSTS. Even if a black-hat hacker poisons the DNS cache, they won’t have the original SSL certificate for your domain, so your users are more likely to spot that they are connected to a phishing website when their browser displays a warning. On the client side, a good option is to use a trusted VPN, where the DNS cache is configured on the VPN side and not on the local router.

## 4 Proof-of-Concept

We chose to use Ettercap for carrying out a DNS Cache Poisoning Attack. The setup includes the victim which in our case is a Windows 10 operating system, the attacker being a Kali Linux virtual machine.

The architecture of this attack is the following:

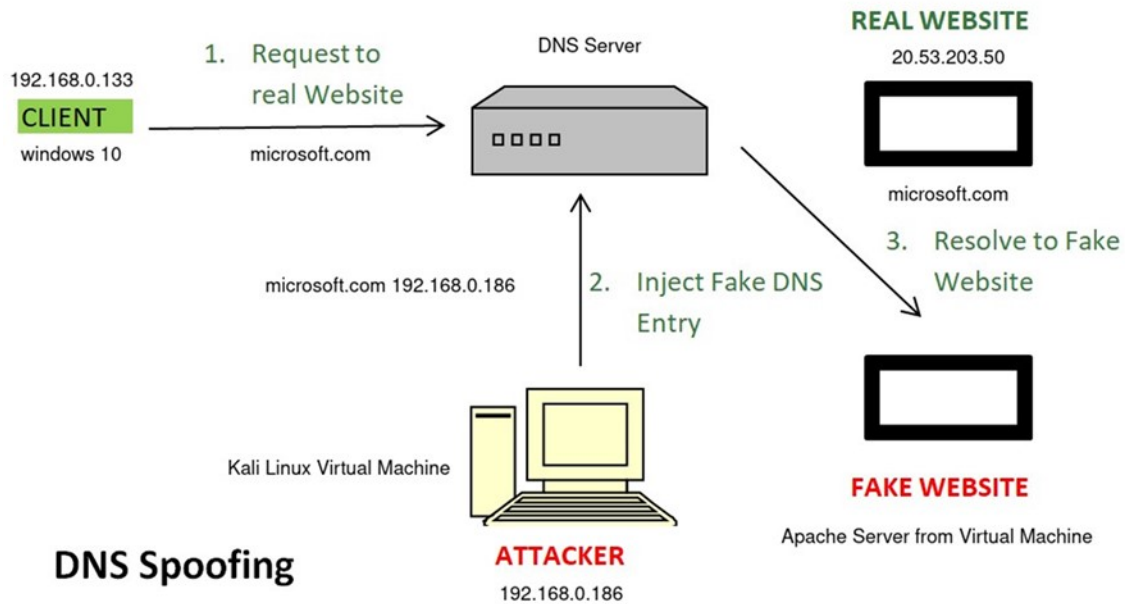
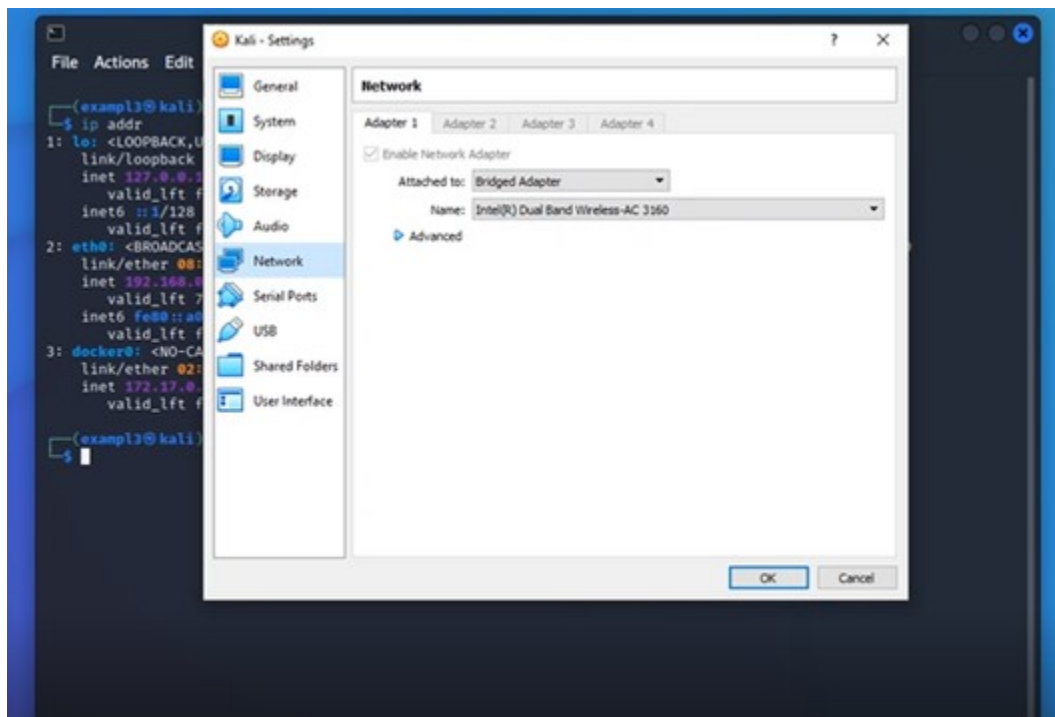


Figure 1: Architecture of Cache poisoning DNS attack

Firstly, we check that the Bridge adapter is enabled and the Apache server is running.




```
(examp13@kali)-[~]
$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
   link/ether 08:00:27:cc:01:e2 brd ff:ff:ff:ff:ff:ff
   inet 192.168.0.186/24 brd 192.168.0.255 scope global dynamic noprefixroute eth0
       valid_lft 7121sec preferred_lft 7121sec
   inet6 fe80::a00:27ff:fecc:1e2/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
   link/ether 02:42:c4:b6:02:11 brd ff:ff:ff:ff:ff:ff
   inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever

(examp13@kali)-[~]
$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; disabled; preset: disabled)
   Active: active (running) since Wed 2023-01-18 07:53:12 EST; 1s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 3305 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 3327 (apache2)
     Tasks: 6 (limit: 4613)
    Memory: 22.4M
       CPU: 370ms
   CGroup: /system.slice/apache2.service
           └─3327 /usr/sbin/apache2 -k start
             3332 /usr/sbin/apache2 -k start
             3333 /usr/sbin/apache2 -k start
             3334 /usr/sbin/apache2 -k start
             3335 /usr/sbin/apache2 -k start
             3336 /usr/sbin/apache2 -k start

Jan 18 07:53:11 kali systemd[1]: Starting The Apache HTTP Server ...
Jan 18 07:53:12 kali apachectl[3321]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name: [192.168.0.186]
Jan 18 07:53:12 kali systemd[1]: Started The Apache HTTP Server.

(examp13@kali)-[~]
$
```



Secondly, we will configure Ettercap files. On etter.conf, gid and uid values will be for both 0, allowing the process to run as root.

```
examp13@kali: /etc/ettercap

File Actions Edit View Help

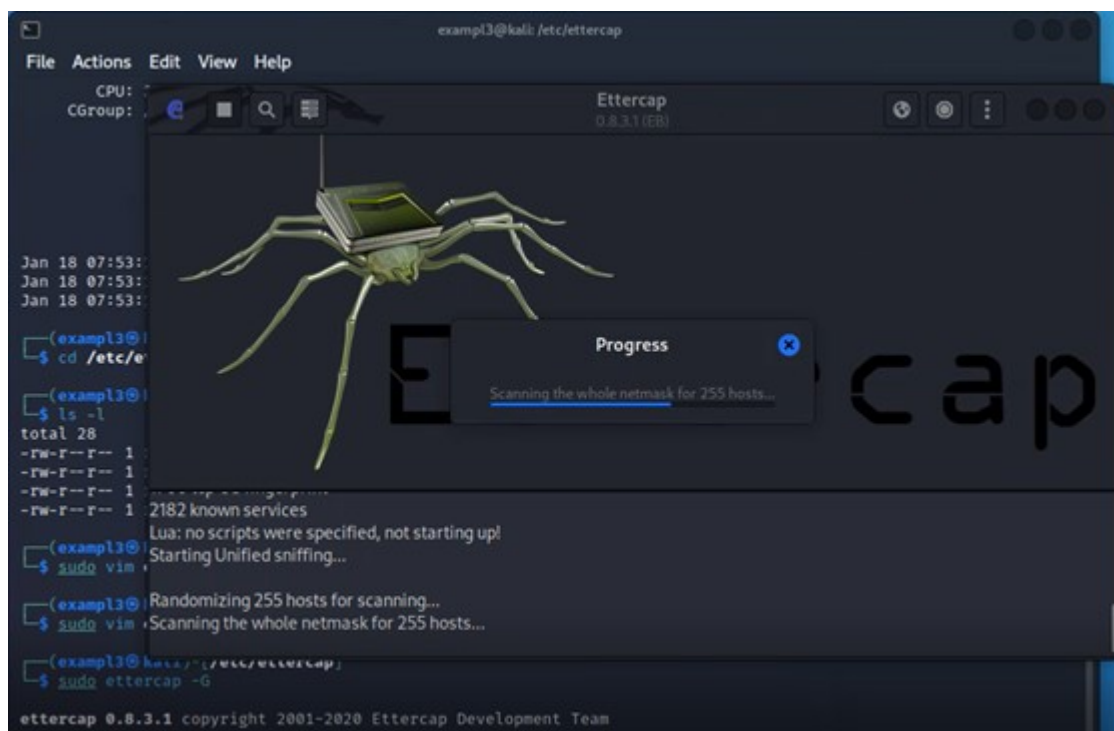
#####
#
# ettercap -- etter.conf -- configuration file
#
# Copyright (C) AloR & NaGA
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the license, or
# (at your option) any later version.
#
#####

[privs]
ec_uid = 0           # nobody is the default
ec_gid = 0           # nobody is the default
```

On the etter.dns we injected our query.

```
example3@kali: /etc/ettercap
File Actions Edit View Help
#####
#
# ettercap -- etter.dns -- host file for dns_spoof plugin
#
# Copyright (C) ALoR & NaGA
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
#####
#
# Sample hosts file for dns_spoof plugin
#
# the format is (for A query):
# www.myhostname.com A 168.11.22.33 3600
# *.foo.com A 168.44.55.66 [optional TTL]
#
# ... for a AAAA query (same hostname allowed):
# www.myhostname.com AAAA 2001:db8::1
# *.foo.com AAAA 2001:db8::2 [optional TTL]
#
# or to skip a protocol family (useful with dual-stack):
# www.hotmail.com AAAA ::
# microsoft.com A 192.168.0.186
# *.microsoft.com A 192.168.0.186
# www.microsoft.com PTR 192.168.0.186
```

The next step is to open Ettercap and scan the hosts present in the network.



It can be seen that our client's IP address is 192.168.0.133.



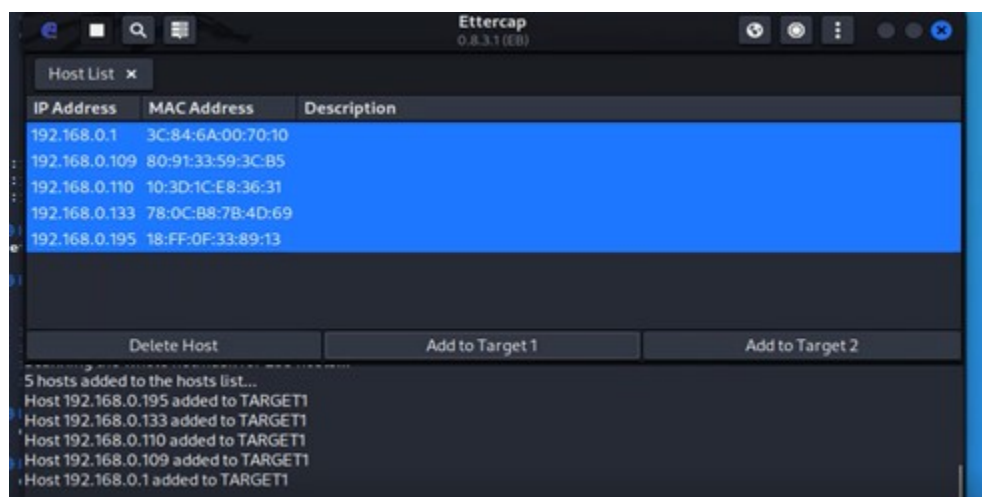
```

Wireless LAN adapter Wi-Fi:

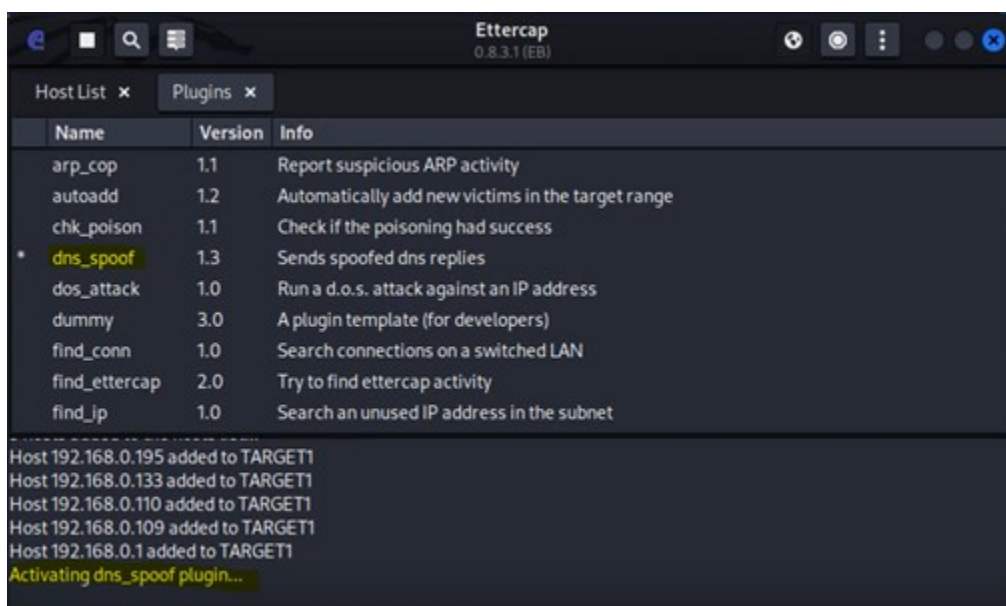
Connection-specific DNS Suffix  . : 
Link-local IPv6 Address . . . . . : fe80::a5a3:f4fd:12be:aed5%19
IPv4 Address. . . . . : 192.168.0.133
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.0.1

```

Then we add the target list for performing the ARP poisoning.

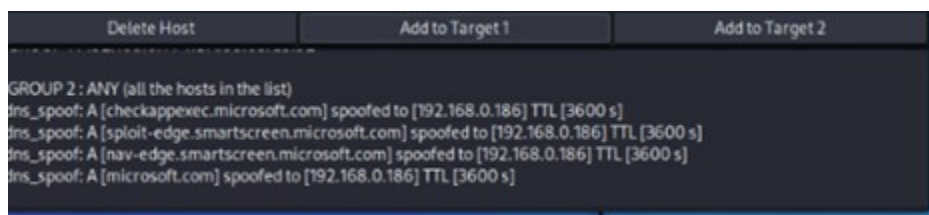


DNS spoof plugin must be enabled also.

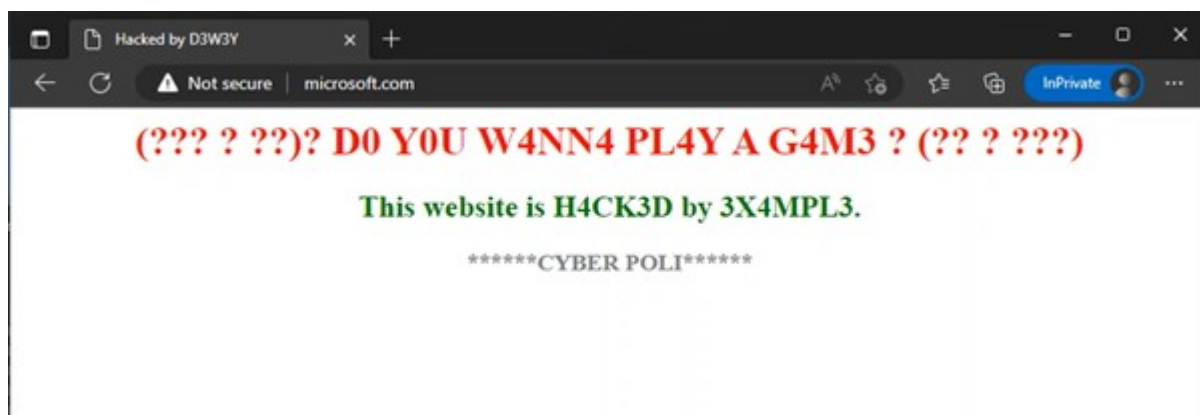


When we open a browser on the victim, we will see a log in the Ettercap window which will prove the spoofing part.





We can see that the attack was a success. When we try on the attacked host to access Microsoft.com domain, we will be redirected to that IP, which in our case will be an Apache server opened on the Virtual Machine.



Ettercap allowed us to listen for DNS requests and send the client back to the address we wanted them to go to instead of the one that would have been returned by the DNS server.

One way to defend against the use of Ettercap by hackers to damage your network security is to scan every endpoint for the Ettercap process. This can easily be performed by any endpoint detection and response (EDR) service, which will probably already be primed to spot and kill Ettercap.

## 5 Results

DNS cache poisoning can be summarized as an attacker controlling the DNS server to send fake DNS responses. As a result, when the user visits the counterfeit domains, they will be directed to a new IP address selected by the hacker.

This new IP address might be from a malicious phishing website, where the users are prompted to download malware, or they might be asked to provide their financial or login details. Also, the risk of installing a keylogger on your computer could cause other sites you visit to be exposed to their usernames and passwords.

Another major danger is that if the website of an internet security company is spoofed, then the device of a user could be exposed to additional threats such as viruses or trojans because there will be no valid security updates.

The real reason why DNS cache poisoning is such an issue is that there is no real way to verify whether the DNS responses you receive are truly genuine or if they have been manipulated.

## 6 References

- [1] Information Security CS 526
- [2] Black Ops 2008: It's The End Of The Cache As We Know It
- [3] X. Zheng et al., "Poison Over Troubled Forwarders: A Cache Poisoning Attack Targeting DNS Forwarding Devices."
- [4] S. Son and V. Shmatikov, "The Hitchhiker's Guide to DNS Cache Poisoning."
- [5] X. Zheng et al., "Open access to the Proceedings of the 29th USENIX Security Symposium is sponsored by USENIX. Poison Over Troubled Forwarders: A Cache Poisoning Attack Targeting DNS Forwarding Devices Poison Over Troubled Forwarders: A Cache Poisoning Attack Targeting DNS Forwarding Devices," 2020.
- [6] "DNS and The Bit 0x20," Hypothetical Me, Jan. 19, 2018
- [7] OWASP, Anatomy of a DNS Cache Poisoning Attack.
- [8] Security Protocols Course. Application Layer Security. Faculty of Automatic Control and Computer Science. Polytechnic University of Bucharest.
- [9] X. Zheng et al., "A Cache Poisoning Attack Targeting DNS Forwarding Devices
- [10] "An Illustrated Guide to the Kaminsky DNS Vulnerability," [unixwiz.net](http://unixwiz.net)