DNS Cache Poisoning: Spoofing Attacks

Members
Casa Vaca Victor David [1]
Garzon Pardo William Alexander [1]

1 Introduction

The Domain Name System (DNS) is a hierarchical naming system that translates domain names, such as www.google.com, into IP addresses, like 8.8.8.8, to facilitate communication with servers. This system was implemented because it is much easier to remember a website's name than to memorize multiple IP addresses to communicate with the destination server. DNS cache is a temporary memory that stores previously made DNS queries on a device to increase response speed to the client. The process is as follows:

First, the DNS resolver is responsible for querying the internal DNS server. If the IP address is in the internal server's cache, the requested page is returned. Otherwise, a request is made to external DNS servers to verify the existence of the IP address and whether the client is authorized to access the site[9]. If the request is successful, the external DNS server returns the server's IP address. This information is also stored in the internal DNS server, along with the domain name, Time to Live (TTL), and the site's IP address. At this point, cache poisoning comes into play, which involves falsifying a DNS query to redirect to a malicious site[8]. Next, we will see an image illustrating how the explanation of cache poisoning works in the DNS cache.

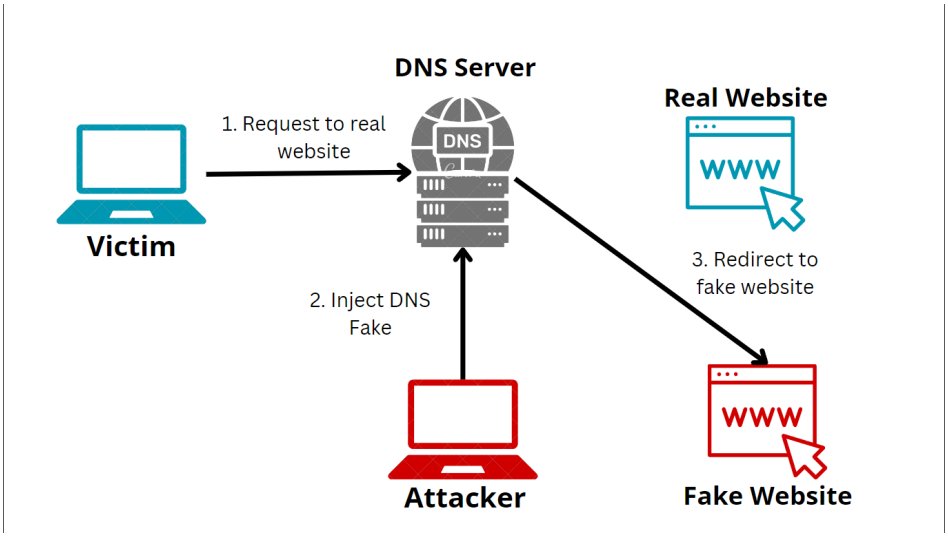


Figure 1: How DNS Cache Poisoning Spoofing Attacks Work

The classic DNS cache poisoning attack targeted a DNS resolver by having an off-path attacker trick a vulnerable DNS resolver into issuing a query to an upstream authoritative name server. Then the attacker attempts to inject rogue responses with the spoofed IP of the name server. If the rogue response arrives before any legitimate ones, and if it matches the "secrets" in the query, then the resolver will accept and cache the rogue results. Specifically, the attacker needs to guess the correct source/destination IP, source/destination port, and the transaction ID (TxID) of the query. In figure 2 we can see how this explanation is graphically explained

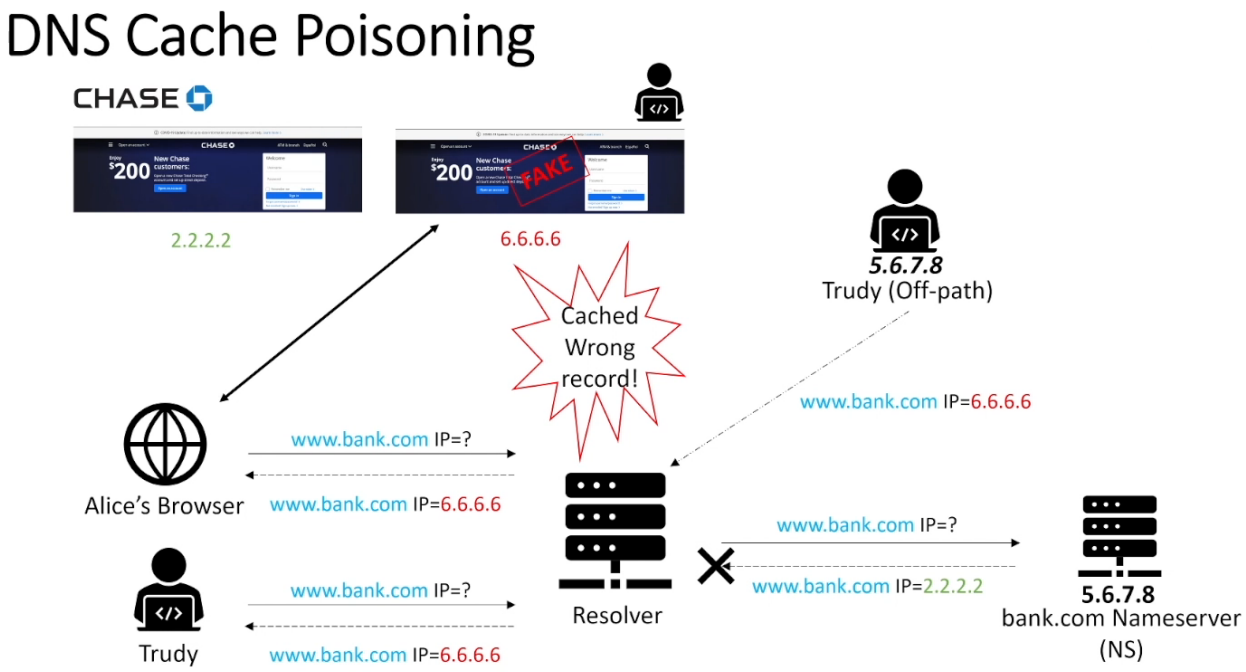


Figure 2: How DNS server works

2 Objetives

To analyze DNS cache poisoning in depth, including its techniques, tools and prevention measures, we have the following plan:

- The different techniques of DNS cache poisoning attack, such as: Cache poisoning attacks by redirection, Cache poisoning attacks by DNS server spoofing, will be studied.
- Try to make a type of attack in a controlated laboratory
- Analyze the tools used to carry out and prevent DNS cache poisoning spoofing attacks
- Identify the most effective security measures to prevent this type of attacks.
- Recommendations will be formulated for the prevention of DNS cache poisoning spoofing
- Additional security measures will be proposed to strengthen the security of computer networks

3 Scope

The project has as its goals the following parameters:

- Collection and analysis of scripts to perform DNS poisoning attacks in Kali Linux, various scripts and tools available in Kali Linux that are used to perform DNS poisoning attacks will be investigated and collected.
- Attack tests using selected scripts on your own DNS server with the goal of achieving effective DNS poisoning. And if so, to try it with external DNS servers.
- Detailed documentation of the attack process, including the results in screenshots.
- Analysis of security measures and mitigation strategies against DNS poisoning.

4 Resources Required

As resources we decided to divide the hardware and software that we will use into 2 resources:

- As hardware we have the use of our computers in which we will use one to raise a web server and the other for testing and testing of the project.
- As software we will use a virtual machine in which we will install Kali Linux since this OS allows us to find several DNS poisoning scripts.

5 Methodology

For this section, we will divide it into 3 parts, which are as follows: create a malicious website, configure our machine as a DNS Server, configure the router to redirect our DNS Server. To create this virtual laboratory environment and carry out practical experiments to analyze vulnerabilities and perform simulations, several key physical and technological resources are required. These include:

- **High-powered machine:** It must allow us to use Kali Linux with all its functions. This operating system will be essential for carrying out the DNS spoofing attack and performing the necessary experiments. Kali Linux offers a wide range of security and penetration testing tools, and we will configure it as our DNS Server.
- **Network router:** To ensure proper and legal handling of the case study, a network router will be used. This device will allow the creation of a controlled and secure local network, in which the DNS spoofing simulations will be carried out. The network will be composed only of the virtual machine acting as the attacker and another machine that will be the victim.
- **Knowledge in networks and security:** Last but not least, solid knowledge in networking and computer security will be needed to properly configure the laboratory environment, carry out the DNS spoofing attacks in a controlled manner, and analyze the obtained results.

5.1 Create a malicious website

One of the main steps in DNS poisoning is the creation of the website to which we are going to redirect the victim. In real attacks, they usually set up a dedicated server specifically to clone a website or social network and connect it to a database where the victims' information is collected. For this scenario, we have decided to use the SEToolkit (Social-Engineer Toolkit) tool.

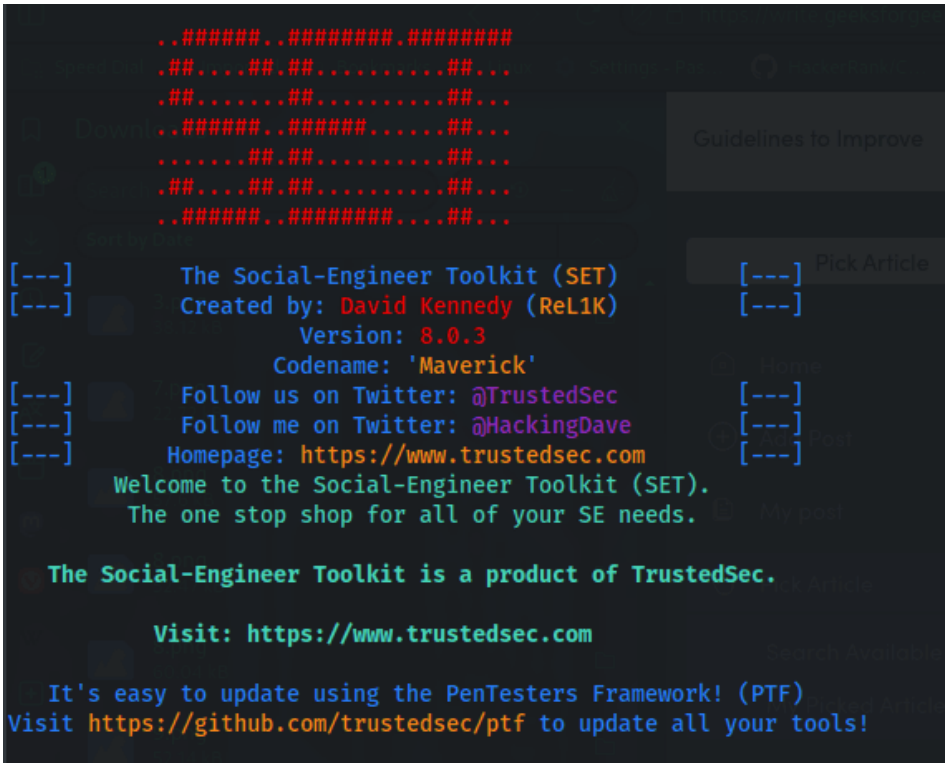


Figure 3: Setoolkit

The Social-Engineer Toolkit is an open-source penetration testing framework designed for social engineering. SET has a number of custom attack vectors that allow you to create a believable attack quickly. SET is a product of TrustedSec, LLC – an information security consulting firm located in Cleveland, Ohio. SEToolkit is a powerful and versatile tool commonly employed by penetration testers and ethical hackers to assess the security of systems and organizations. However, it's essential to understand the potential risks and vulnerabilities associated with these techniques to better protect against them. In our laboratory and for educational purposes to demonstrate the potential of these tools, we have decided to clone the Facebook and Instagram websites. Additionally, an important aspect of this tool is that it returns the information entered by the victim in plain text. For other types of sites, the process would be the same for all cases, as this tool has a large number of website templates that can be cloned.

5.2 Procces to configure our machine as a DNS server

When it comes to configuring our machine as a DNS server, it is of utmost importance to correctly apply the settings. For this lab, we will use the BIND 9 tool, a widely used open source DNS server software. BIND 9 performs key functions, such as serving as a DNS server to translate domain names to IP addresses, caching DNS information to improve performance, enabling DNS zone management, providing security through DNSSEC, and performing recursive queries and forwarding queries to other servers. This tool offers flexibility and scalability to meet the needs of the Internet infrastructure.

Once this tool is installed, we will proceed to modify the ***named.conf.local*** file. This file defines the local DNS zones that will be managed by this BIND server. Information such as the zone name, the location of the zone files and the type of zone (master, slave, forwarding, etc.) is specified here. Below is an example of how this file can be viewed:

```
1
2 ## Forward Zone
3 zone "Dawills.com" IN {
```

```
    type master;
    file "/etc/bind/zones/Dawills.com.db";
        allow-query{ any;};
        allow-transfer {192.168.1.16;};
};
## Reverse Zone
zone "1.168.192.in-addr.arpa" IN {
    type master;
    file "/etc/bind/reverse/1.168.192.in-addr.arpa";
        allow-query{ any;};
        allow-transfer {192.168.1.16;};
};
```

In this file, we define the locations of our zone configurations, which are crucial for caching the domain we register and achieving identity spoofing.

5.2.1 Direct Zone

This section defines the direct zone for the domain `Dawills.com`.

- The `type master` directive indicates that this server is the primary (authoritative) server for this domain.
- The `file "/etc/bind/zones/Dawills.com.db"` directive specifies the location of the zone file, which contains the actual DNS records for the domain.
- The `allow-query { any; }` directive allows any client to query the DNS server for this domain.
- The `allow-transfer {192.168.1.16;}` directive allows the DNS server with IP address `192.168.1.16` to transfer the zone file.

5.2.2 Reverse Zone

This section defines the reverse zone for the network `1.168.192.in-addr.arpa`.

- The `type master` directive indicates that this server is the primary (authoritative) server for this reverse zone.
- The `file "/etc/bind/reverse/1.168.192.in-addr.arpa"` directive specifies the location of the reverse zone file, which contains the reverse DNS records for the network `192.168.1.16/24`.
- The `allow-query { any; }` directive allows any client to query the DNS server for this reverse zone.
- The `allow-transfer {192.168.1.16;}` directive allows the DNS server with IP address `192.168.1.16` to transfer the reverse zone file.

The server allows any client to query the DNS records of these zones and permits the DNS server at `192.168.1.16` to transfer the zone files.

Now we should create the zone that previously in the previous file we placed the location of these zones, first we will begin with the direct zone that in this case is sudo **dawills.com.db** the configuration of this zone would look more or less like this:

```
;
; Zone file for the domain dawills.com
;
$TTL    86400   ; Default TTL of 24 hours
@       IN      SOA     ns1.dawills.com. admin.dawills.com. (
                        2023060301      ; Serial number (yyyymmddxx)
                        3600            ; Refresh time (1 hour)
                        600             ; Retry time (10 minutes)
                        86400           ; Expiration time (1 day)
```

```
                        3600 )          ; Minimum TTL (1 hour)

                IN      NS      ns1.dawills.com.
                IN      NS      ns2.dawills.com.

; A records for the hosts
ns1             IN      A       192.168.1.100
ns2             IN      A       192.168.1.101
www             IN      A       192.168.1.102
mail            IN      A       192.168.1.103

; MX record for the mail server
mail            IN      MX      10 mail.dawills.com.
```

The dawills.com.db file contains the direct zone configuration for the dawills.com domain. This file defines the DNS records required for this domain, such as the A records for the hosts and the MX record for the mail server. Important parameters such as the default time to live (TTL) and the SOA (Start of Authority) values including the serial number and the refresh, retry, expiration and minimum TTL times are also set.

Now we will go with the inverse zone that is the one that will be in charge of returning the answers in our case the file is the following **1.168.192.in-addr.arpa**

```
;
; Reverse zone file for the 192.168.0.0/24 network block
;
$TTL    86400
@       IN      SOA     ns1.dawills.com. admin.dawills.com. (
                        2023060301      ; Serial number
                        3600            ; Refresh time
                        600             ; Retry time
                        86400           ; Expire time
                        3600 )          ; Minimum TTL

                IN      NS      ns1.dawills.com.
                IN      NS      ns2.dawills.com.

; PTR records for the IP addresses
100             IN      PTR     ns1.dawills.com.
101             IN      PTR     ns2.dawills.com.
102             IN      PTR     www.dawills.com.
103             IN      PTR     mail.dawills.com.
```

The network block represented in reverse format: 0.168.192.in-addr.arpa. The SOA (Start of Authority) parameters with the serial number, refresh time, retry time, expire time, and minimum TTL. The NS (Name Server) records pointing to the primary and secondary name servers. The PTR (Pointer) records that map the IP addresses to the corresponding hostnames.

Once the zones are configured now we must modify the following ***named.conf.options*** file in this file will help us to create the addressing to our previously created zones and will tell you that you must always first look for the domain in our zone and if you do not find it decide to ask an external DNS server such as google our code looks similar to:

```
acl "trusted" { #An ACL directive that defines our local area network
        192.168.1.0/24;
        192.168.1.16;
        192.168.1.17;
        };
options {
        directory "/var/cache/bind";
        recursion yes;
        allow-recursion {
                                trusted;
                        };
        allow-query {
                                trusted;
                    };
```

```
        listen-on {
                        192.168.1.16; 192.168.1.17;
                };
        allow-transfer {
                                trusted;
                };
        forwarders {
                        4.2.2.4;
                };

        dnssec-validation auto;

        listen-on-v6 { any; };
};
```

ACL (Access Control List) Directive block

- `acl "trusted" { ... };`: Defines an access control list (ACL) named "trusted" that includes the specified IP addresses and network range.
- `192.168.1.0/24;`: Includes the entire subnet 192.168.1.0 to 192.168.1.255. This means that any device with an IP address in this range is considered part of the "trusted" network.
- `192.168.1.16;`: Specifically includes the IP address 192.168.1.16.
- `192.168.1.17;`: Specifically includes the IP address 192.168.1.17.

Options Block

- `directory "/var/cache/bind";`: Specifies the working directory for the BIND DNS server where it will store its cache files. This is where the server temporarily saves information to speed up responses to future queries.
- `recursion yes;`: Enables recursive queries on the DNS server, meaning it will query other DNS servers on behalf of the client to resolve domain names it does not know directly.
- `allow-recursion { trusted; };`: Restricts recursive queries to only clients within the "trusted" ACL. This helps prevent abuse of the DNS server by unauthorized users.
- `allow-query { trusted; };`: Restricts general DNS queries to only clients within the "trusted" ACL. This ensures that only trusted devices can perform DNS queries on the server.
- `listen-on { 192.168.1.16; 192.168.1.17; };`: Configures the DNS server to listen for queries on the specified IP addresses 192.168.1.16 and 192.168.1.17. This means the server will only respond to requests that come to these IP addresses.
- `allow-transfer { trusted; };`: Restricts zone transfers (used to replicate DNS databases between servers) to only clients within the "trusted" ACL. This helps protect sensitive DNS information.
- `forwarders { 4.2.2.4; };`: Specifies an upstream DNS server (in this case, 4.2.2.4) to which queries will be forwarded if the DNS server cannot resolve them locally. This allows the DNS server to rely on another server for answers.
- `dnssec-validation auto;`: Enables automatic DNSSEC validation, which helps ensure the authenticity and integrity of DNS data. DNSSEC (Domain Name System Security Extensions) adds digital signatures to DNS data.
- `listen-on-v6 { any; };`: Configures the DNS server to listen for IPv6 queries on all available interfaces. This means the server is ready to handle requests from both IPv4 and IPv6 clients.

Finally we must apply all our configurations and make sure that there are no errors by executing the following command lines:

- ***named-checkconf /etc/bind/named.conf:*** Check the syntax of the main BIND configuration file to ensure that there are no errors. Then we proceed to run the resolver as follows:

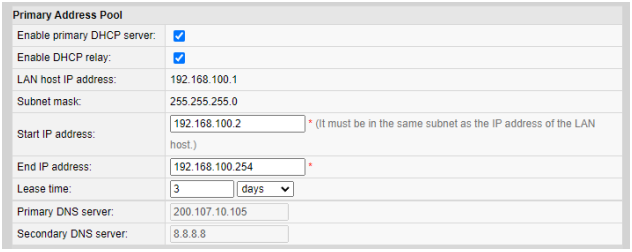
- ***sudo nano /etc/resolvconf/resolv.conf.d/head:*** In this file we have to put our IP address that will be the one of our DNS server to change the direction of the traffic and finally we execute:

- ***sudo resolvconf -u:*** Updates the resolv.conf file with the changes made in the resolvconf configuration files.

To verify that everything has gone correctly, we must check that the proper routing exists. To do this, we can use the nslookup command as follows:
nslookup dawills.com.
If the response we get is the IP address of our DNS server, we can conclude that the redirection has been successful. We should get a result similar to this:

Server: 192.168.1.16
Address: 192.168.1.1653

In this case, we can see that the operation has been successful. Finally, the last step is to configure the router so that, via DHCP, it sets our DNS server with the IP address of our machine previously configured. This will ensure that all DNS query traffic from people connecting to the router will pass through our machine, ensuring that victims are redirected to our malicious page. In this way, the DNS poisoning attack at the LAN level will be successful. Next we can see a configuration of the router where we set the main DNS server to be 200.107.10.105 and here is where we must modify it and put the address of our machine



6 Results and Discussion

To test all the steps mentioned above, we decided to create a controlled laboratory environment. As a test, we used the social network Facebook. The first thing we did was to use the SET Toolkit tool, and as shown in Figure 4, we successfully cloned the Facebook page. The only way to realize that it is actually the cloned site is by looking at the URL, since it is not a secure site and also shows an IP address. This IP address is the one we will use to redirect victims when they enter “facebook.com” in the URL and are on the LAN we set up in this lab.

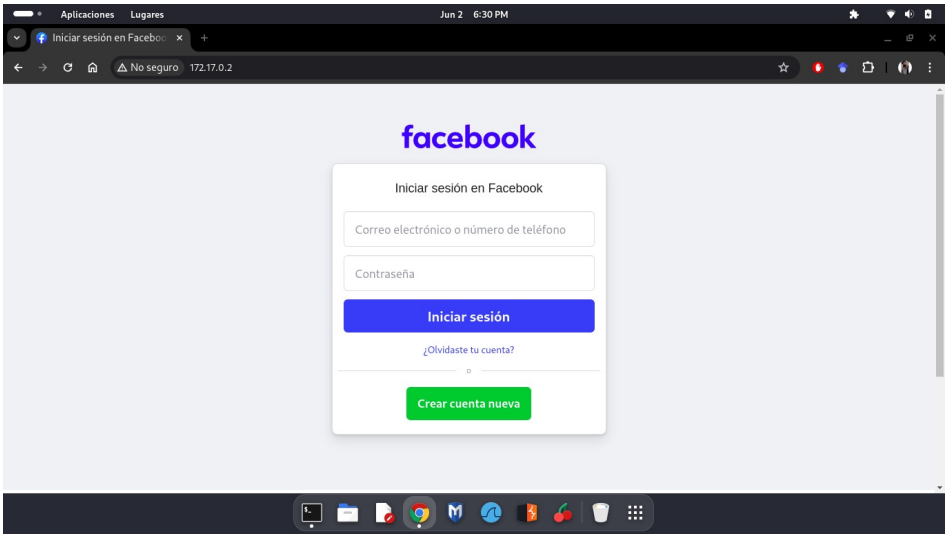


Figure 4: Clone site web

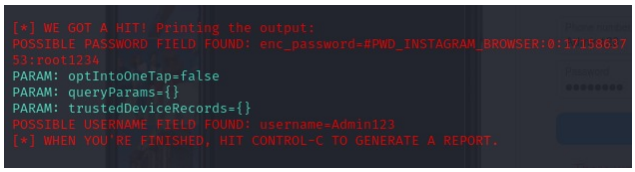


Figure 5: Credential stored

We tested whether our cloned website stores the information that the victim enters. As can be seen in Figure 5, the credential storage was successful.

Then, we proceeded to create the zones and configure our machine as a DNS server, as explained in the previous section. We created the "Facebook.com" and "Facebookk.com" zones in order to test how our DNS poisoning works with a real and a fake domain name. Once both zones were created, we decided to test them using **nslookup**, as shown in Figure 6. In the figure, we can see that our addressing is successful and redirects us to the IP address of our cloned page.

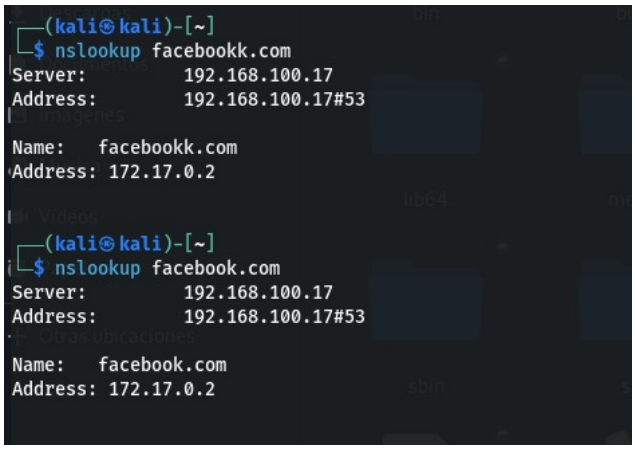


Figure 6: DNS queries of facebook.com and facebookk.com using nslookup tool

Finally, we confirmed that the DNS poisoning was successful and proceeded to configure the router and perform DNS poisoning tests from other machines. The results we obtained when using a web browser from another device were as follows: when typing “Facebook.com”, we obtained what is shown in Figure 7, and when typing “facebookk.com”, we could observe what is shown in Figure 8.

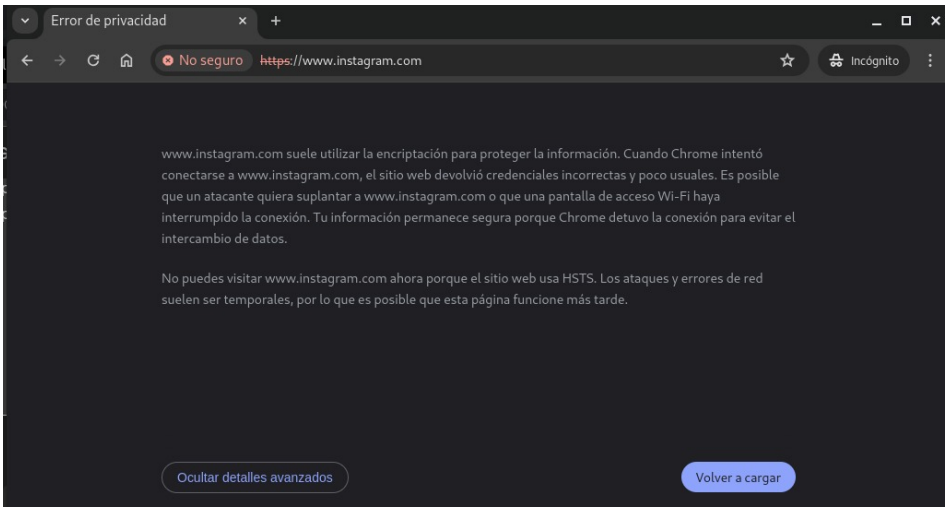


Figure 7: Facebook.com after DNS Poisoning from google chrome on a device discovered on the same LAN

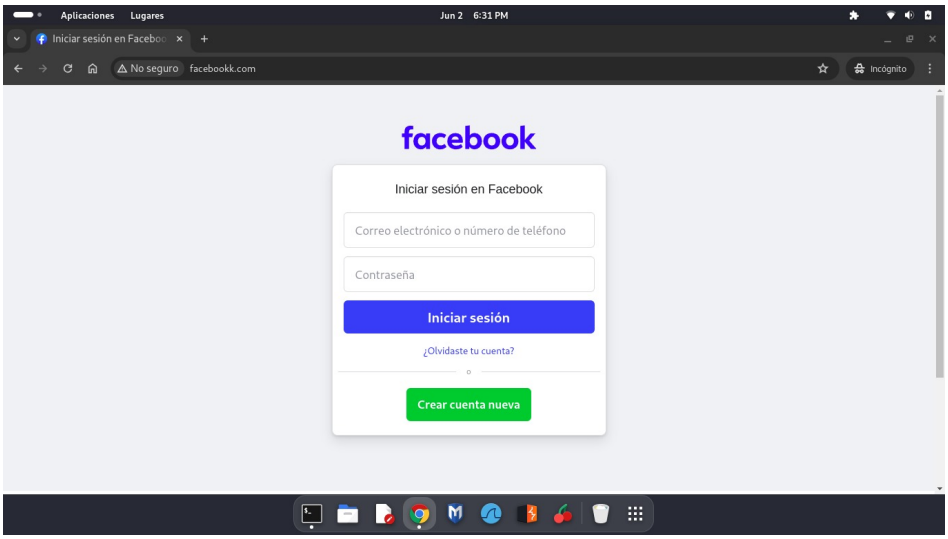


Figure 8: Facebookk.com after DNS Posoning from google chrome on a device discovered on the same LAN

Why did this happen? As we can see, the DNS poisoning worked correctly only for a fake domain name like ”facebookk.com”, but why didn’t it work for an existing, real domain name like ”Facebook.com”? The answer lies in another countermeasure we did not consider: the HSTS protocol. HSTS, or ”HTTP Strict Transport Security”, is a web security policy mechanism that protects websites against man-in-the-middle attacks by forcing connections through HTTPS instead of HTTP. By enabling HSTS, a site instructs browsers to only access it through HTTPS, even if the user tries to access it through HTTP. This prevents attacks like SSL stripping.

In our case, since our web server is an HTTP site, the browser queries the digital signature of ”Facebook.com”

and realizes it is incorrect. This is why it blocks our access, and we cannot access the cloned website due to this restriction. Based on our research, we discovered that this type of measure is applied on the web server side, not the DNS server side, so we could not bypass this countermeasure as it would involve a different type of attack. However, we found that we were still able to launch a DDoS attack on this website, as no device on the network could access "Facebook.com" due to this security measure.

Returning to the main focus of DNS poisoning, we can say that the objective was achieved to a certain extent, as if a user makes a typographical error without realizing it, such as adding an extra "k", they could be redirected to our malicious site, and the real website would be offline. Another way a user could fall into our trap is if we place our fake DNS in a QR code or hyperlink, where the redirection would be to our DNS server, thus succeeding in the attack. How could this attack method be countered, even when the real page appears to be down? After investigating and analyzing how to mitigate it, we came up with the idea that the best way to avoid this attack locally is by using a VPN. If we consider how a VPN works, it essentially redirects our traffic to another location in the world, so our DNS queries no longer go through the configured DNS server and instead go outside, countering this attack method. An example to better understand how VPN works is illustrated in the figure 9 where we can see that it is like a swich where practically our DNS server is modified and becomes another located in another part of the world and our router and server are disconnected and redirected to a secure server.

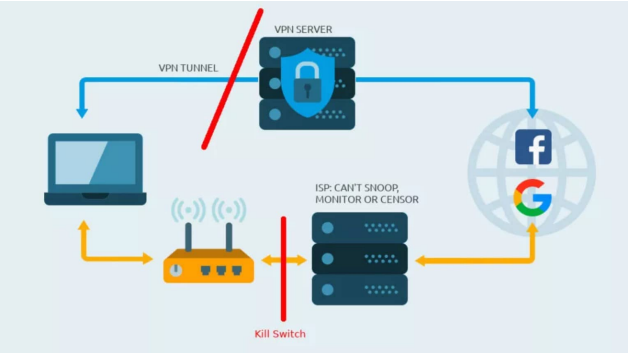


Figure 9: How VPN works

Additionally, the VPN provides a secure tunnel for Internet traffic, making it difficult for attackers to intercept communications. This adds an additional layer of security by preventing DNS traffic from being compromised in the first place. Therefore, using a VPN is an effective measure to protect against DNS poisoning and other man-in-the-middle attacks.

Finally, while a VPN can be an effective solution, it is important to remember that no security measure is foolproof. Therefore, it is essential to stay up to date on security best practices and continue to use other security layers, such as antivirus software and firewalls, to protect against a variety of online threats.

7 Conclusions

- **Verify links before accessing websites:** It is crucial to carefully verify links before clicking on them to avoid being redirected to malicious websites that may compromise our information. This is because phishing attacks and DNS poisoning can trick us into entering confidential information on fake sites. By verifying links, we can identify potential unwanted redirects and protect our online security.
- **Avoid public Wi-Fi networks:** It is advisable to avoid connecting to public Wi-Fi networks, as we do not know who manages them and could be exposed to attacks similar to DNS poisoning. These networks can be vulnerable to man-in-the-middle attacks and other security threats, compromising our private information. By staying away from these networks, we reduce the risk of falling victim to malicious attacks and protect our online privacy.

- **Use of a VPN:** Using a VPN can be an effective countermeasure against DNS poisoning attacks, as it helps change our DNS server and protect our DNS queries from possible manipulations. Additionally, a VPN encrypts our Internet traffic, making it difficult for attackers to intercept our communications and access our confidential information. By using a VPN, we enhance our online security and reduce the likelihood of being victims of DNS poisoning attacks or other security threats.

Finally we give access to a github repository where you can find all the configuration we made of our DNS server: https://github.com/davichocv18/DNS-poisoning

References

[1] N. Tripathi, M. Swarnkar and N. Hubballi, "DNS spoofing in local networks made easy," 2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Bhubaneswar, India, 2017, pp. 1-6, doi: 10.1109/ANTS.2017.8384122.

[2] Steinhoff, U., Wiesmaier, A., and Araújo, R. (2006, June). The state of the art in DNS spoofing. In Proc. 4th Intl. Conf. Applied Cryptography and Network Security (ACNS).

[3] A. A. Maksutov, I. A. Cherepanov and M. S. Alekseev, "Detection and prevention of DNS spoofing attacks," 2017 Siberian Symposium on Data Science and Engineering (SSDSE), Novosibirsk, Russia, 2017, pp. 84-87, doi: 10.1109/SSDSE.2017.8071970.

[4] Spoofing. (2023, Agosto 16). Recuperado el 10 de marzo de 2024, de Malwarebytes website: https://www.malwarebytes.com/spoofing

[5] Dissanayake, I. M. M. (2018, October). DNS cache poisoning: A review on its technique and countermeasures. In 2018 National Information Technology Conference (NITC) (pp. 1-6). IEEE.

[6] Trostle, J., Van Besien, B., and Pujari, A. (2010, October). Protecting against DNS cache poisoning attacks. In 2010 6th IEEE Workshop on Secure Network Protocols (pp. 25-30). IEEE.

[7] Hussain, M. A., Jin, H., Hussien, Z. A., Abduljabbar, Z. A., Abbdal, S. H., and Ibrahim, A. (2016, July). DNS protection against spoofing and poisoning attacks. In 2016 3rd International Conference on Information Science and Control Engineering (ICISCE) (pp. 1308-1312). IEEE.

[8] Olzak, T. (2006). Dns cache poisoning: Definition and prevention. *Disponıvel em http://adventuresinsecurity. com/Papers/DNS_Cache_Poisoning. pdf.*

[9] Zhao, Y., Hu, N., Zhang, C., & Cheng, X. (2020). DCG: A Client-side Protection Method for DNS Cache. *J. Internet Serv. Inf. Secur., 10*(2), 103-121.