Department of Information Engineering and Computer Science (DISI)
University of Trento, Italy

Network Security - AY 2017/2018

Lab Activity Report

Man in the Middle attacks

Authors:
Gabriele Gemmi [198042]
Lorenzo Brugnera [197054]

1 Introduction

What are Man in the Middle (MitM) attacks In a MitM attack a malicious user secretly relays and possibly alters the communication between two parties who believe they are directly communicating with each other. The aim of the attacker is to take the control of the communication and to intercept all relevant messages passing between the two victims.
Once the attacker puts himself "*in the middle*", he can do eavesdropping, which means secretely listening to a private conversation of other parties without their consent. This type of attack is very effective when the connection between the two victims is not secure (HTTP). In this way, all the traffic is exchanged in clear and the attacker can easily manage to intercept sensitive data, like usernames, passwords, cookies and so on.
A MitM attack can be performed on secure connections as well, with the only difference that the attacker has to establish two independent SSL sessions, one for each TCP connection. Generally, browsers handle this kind of things, notifying the user that the certificate is not valid. A MitM is well represented by the figure 1. There are several ways to mount a MitM, but one important requisite is that the attacker must be "close" to the victims.

1.1 Network attacks

- ARP poisoning
- DHCP (DHCPv6) poisoning
- Evil Twin

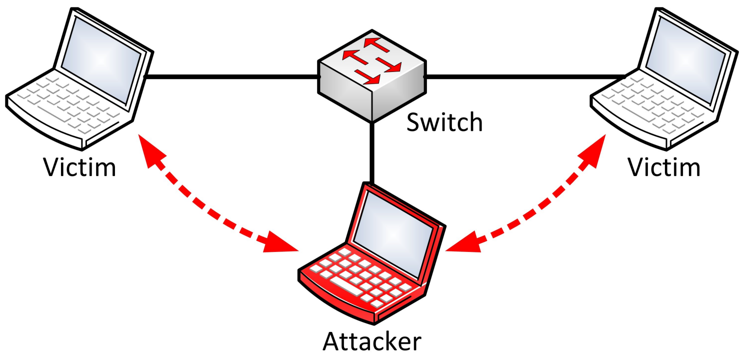

Figure 1: Diagram of a MitM attack

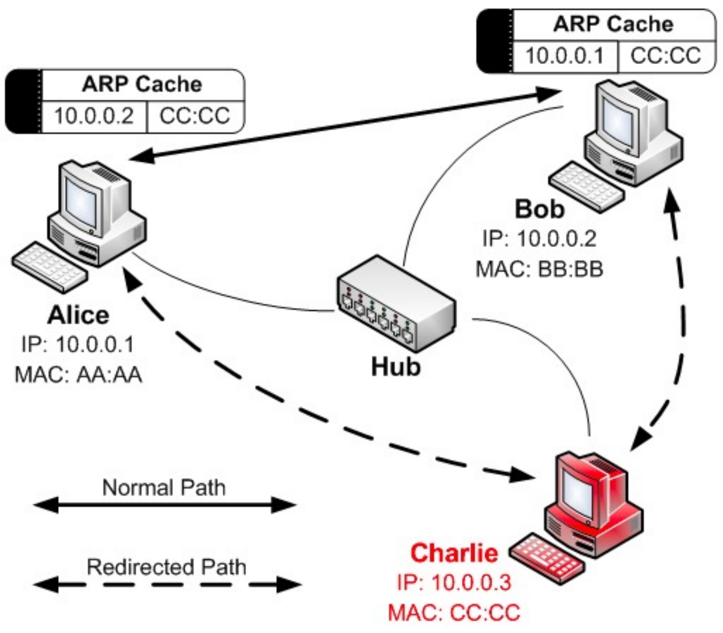

Figure 2: ARP Spoofing attack diagram

1.2 Service attacks

1.2.1 ARP Poisoning

The Address Resolution Protocol (ARP) is a communication protocol used to discover link layer addresses, such as MAC addresses, associated with a given network layer address, which is typically IPv4. This mapping between MAC and IP addresses is a critical function in the Internet protocol suite.
ARP poisoning is a technique by which an attacker sends spoofed ARP messages onto a local area network with the purpose to alter the mapping between IP and MAC addresses. It may allow an attacker to intercept data frames on a network, sniffing sensitive information or altering the traffic. Often the attack is used as an opening for other attacks, such as denial of service, man in the middle, or session hijacking. The basic principle behind it is the exploitation of the the ARP protocol by sending spoofed messages onto the LAN. This attack can be run from a compromised host on the LAN, or from an attacker's machine that is connected directly to the target LAN.
Generally, the goal of the attack is to associate the attacker's MAC address with the IP address of a victim, so that any traffic meant for the victim will be sent to the attacker. The diagram visible in figure 2 shows the behaviour of the attack.

Countermeasures Several countermeasures can be adopted to prevent an ARP poisoning attack:

- **ARP poisoning proof switches**. These smart switches maintain an ARP table and blocks illegitimate ARP responses.
- **Port-Based Network Access Control (802.1x)**. This protocol provides a mechanism to authenticate devices which want to establish a connection to LAN or WLAN network through a switch or a Wi-Fi access point.
- **Encrypted VPN**. These VPNs authenticate both client and server with a public key mechanism and so the attacker cannot intercept nor alter the communication.

1.2.2 DHCP Poisoning

Dynamic Host Configuration Protocol (DHCP) is a network management protocol used in networks to dynamically assign IP address to each device. In this way, the user does not need to configure manually its device, that works in a plug and play fashion.
DHCP poisoning is a technique which consists in setting up a rogue DHCP server controlled by the attacker. Each time a client sends a DHCP request, the rogue's response is supposed to arrive earlier than the real one. When the legit response arrives, it is simply discarded. The rogue server can advertise a fake gateway to the clients, in this way all the traffic going away from the local subnet will start to flow into the attacker machine. Then, the attacker forwards the received traffic accordingly, so the clients will not notice any disruption in connectivity. The diagram in figure 3 explains how the attack works.

Countermeasures Smart switches are one of the best solutions to avoid this kind of attack. They are based on a mechanism which sets a port as trusted or untrusted. As shown in the figure 4, the router expects to receive DHCP responses only from the real DHCP server, so all the responses which do not come from the legit server are discarded.

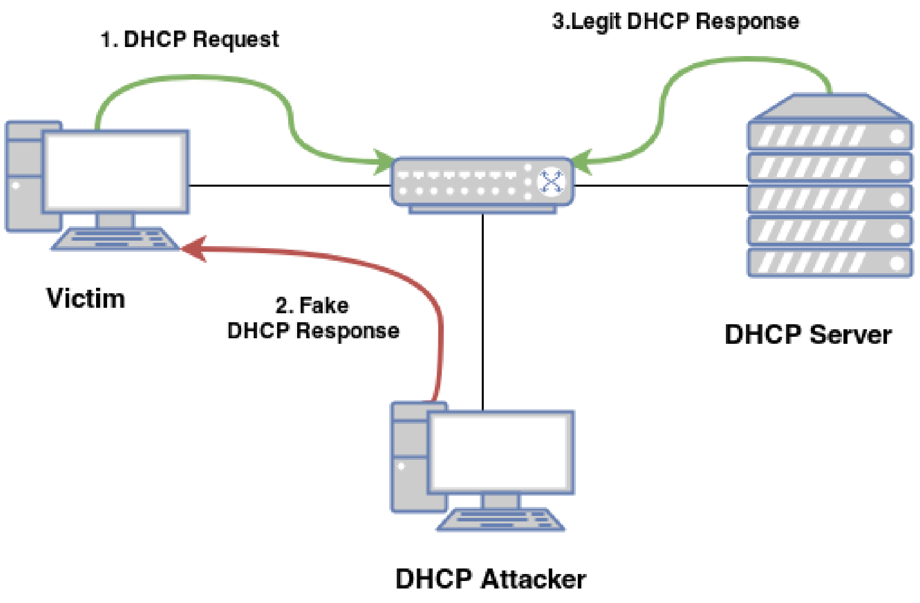

Figure 3: DHCP poisoning attack diagram

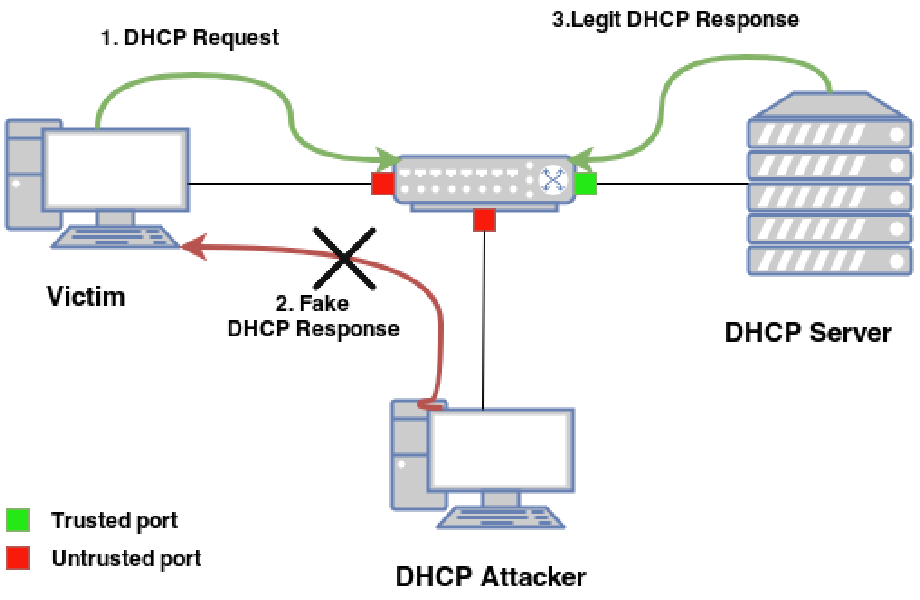

Figure 4: DHCP snooping diagram

1.2.3 Evil Twin

The 802.11[1] protocol allows the creation of a wireless network with multiple Access Points (Extended Service Set). This functionality is used in large environments where more than a single AP is needed. In order to allow devices to perform handover, multiple APs, sharing the same Service Set IDentifier (SSID), are set up.
An Evil Twin is a rouge AP added to an existing ESS to eavesdrop the communication.
The attacker sets up the rouge AP close to the victim that will receive it with a stronger signal than the legitimate one. The diagram in figure 5 explains the attack.

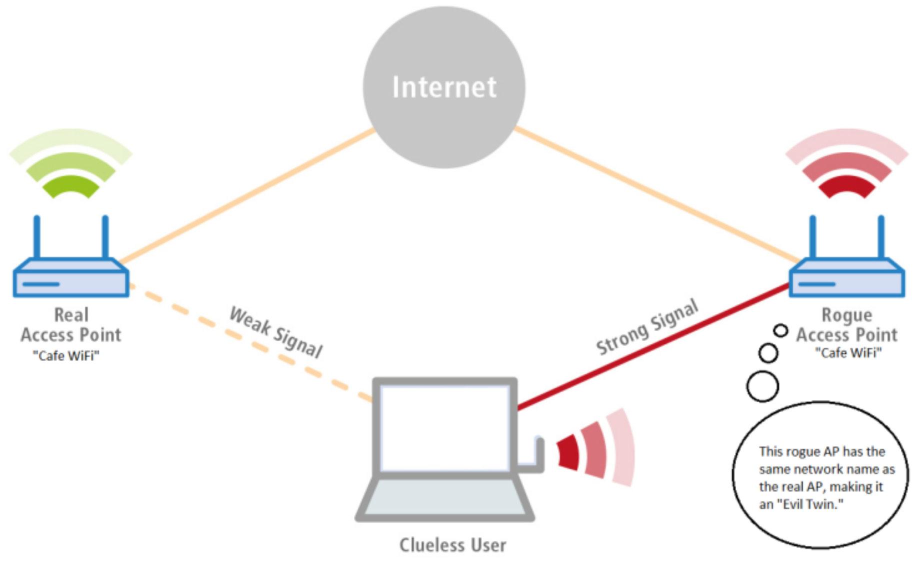

Figure 5: Evil Twin attack diagram

Countermeasure Usually, a simple authentication (WPA) does not ensure the client that the AP is the legitimate one, because the attacker just need to discover the key. Therefore, to prevent this attack, the client must verify the legitimacy of the AP. The 802.1x protocol implements this policy using Public Key Infrastructure (PKI).

[1]`https://tools.ietf.org/html/rfc7494`

2 Laboratory activity overview

The following diagram shows the network topology we set up in order to mount all the attacks we wanted to explain. It is quite simple and composed by:

- **Victim (Client)**. A Xubuntu machine where alumns simulate the behaviour of a victim visiting web pages on a browser.
- **Server**. A Linux machine running Apache HTTP Server that represents *Internet*.
- **Attacker**. A Xubuntu machine used by alumns to mount the attacks.
- **Router**. An OpenWRT router used to interconnect the previous machines. It is also used as a DNS server.

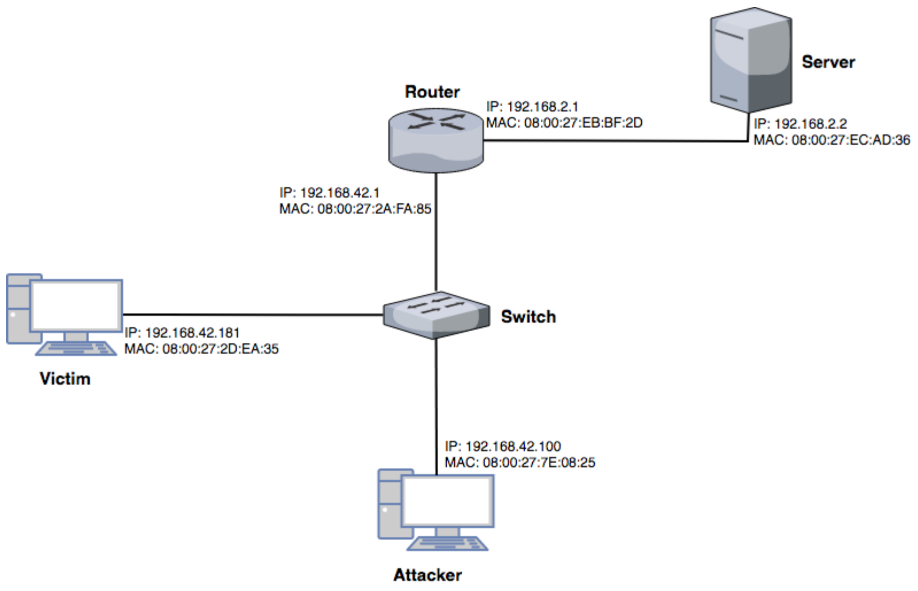

Figure 6: Topology of the VMs network

3 Exercises

3.1 Exercise 1 - HTTP Interception

This preparatory exercise is used to let alumns build confidence with the machines. Basically, in this part alumns have to capture all the traffic that flows between the victim and the router. Since the traffic is not encrypted, sensitive information are sniffed by the attacker.
First of all alumns have to mount a Man in the Middle (MitM) network attack. Theoretically any of the attacks explained before could be used. For simplicity we chose to mount an ARP Spoofing attack using this simple command line tool available in the dsniff package[2]:

```
arpspoof -t <victim_ip>  -r <router_ip>
```

After this, from the client virtual machine, users have to navigate to a non-secure webpage (`http://www.homepage.it`) and, then, using Wireshark[3] sniff the HTTP traffic exchanged between the victim and the server.

3.1.1 Countermeasure

SSL/TLS is a cryptographic protocol aimed at preserving privacy and data integrity over a computer network between two communicating computer applications. HTTPS is the secure extension of HTTP implementing SSL/TLS.

[2]`https://www.monkey.org/~dugsong/dsniff/`
[3]`https://www.wireshark.org/`

3.2 Exercise 2 - SSL Stripping

Roughly speaking, SSL Strip is a technique by which every occurrence of `https://` webpage gets replaced in `http://`. In this type of attack all the traffic from the victim's machine is routed into the attacker's one via a transparent proxy (sslstrip[4]) that is created by the attacker himself.
The interesting thing is that the victim will not get any notification by the browser regarding SSL certificate errors. The figure 7 shows that the client is supposed to directly communicate with the server, whereas all the traffic is forwarded to the attacker's machine and, then, to the server. The diagram displayed in figure 8 better explains all the temporal sequence of the attack.

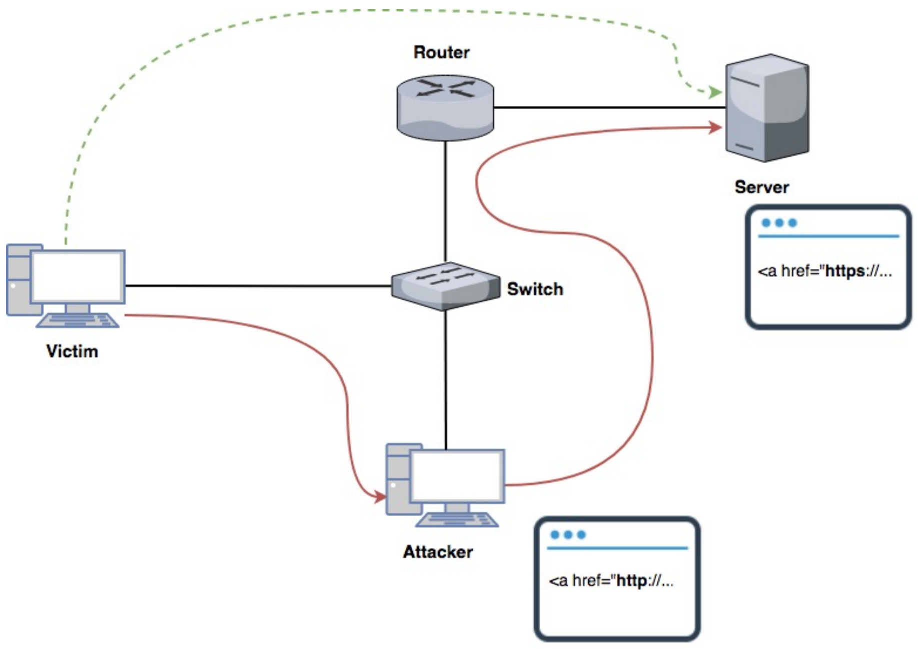

Figure 7: SSL Stripping attack diagram

1. The user agent makes a request for a non-secure webpage
2. All the traffic reaches the attacker's virtual machine
3. Using iptables the attacker redirects all the HTTP traffic from the port 80 to the port where sslstrip is running
4. The HTTP traffic is forwarded to the server and the response is sent to sslstrip

[4] `https://moxie.org/software/sslstrip/`

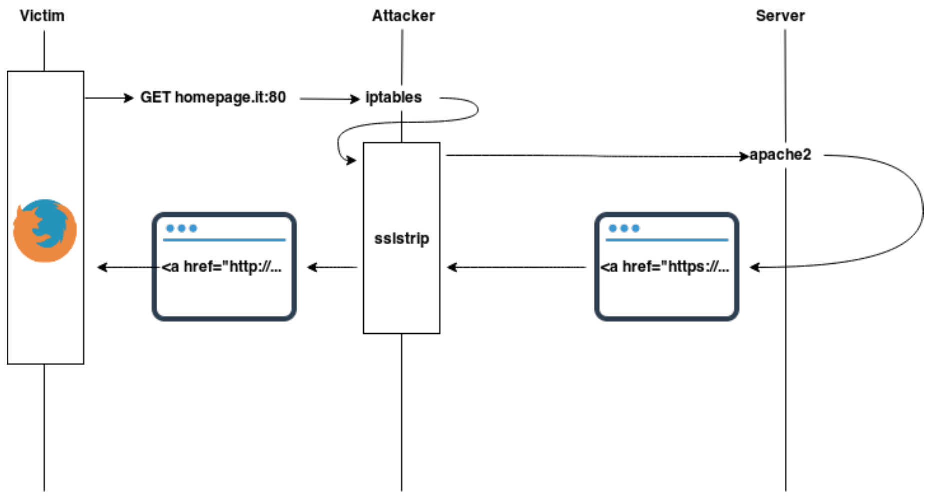

Figure 8: SSL Stripping attack diagram

5. Sslstrip downgrades each `https://` link into an `http://` one
6. The stripped webpage is returned to the user agent

To properly mount this attack, the alumn have to follow this procedure:

- Mount a MitM attack
- Setup sslstrip to manipulate the HTTP traffic using this command line tool: `sslstrip -l <port>`
- Using iptables, create a specific rule to redirect the traffic from the port 80 to the port where sslstrip is running:
```
sudo iptables -t nat -A PREROUTING -p tcp --destination-port 80
-j REDIRECT --to-port <sslstrip_port>
```
- Navigate to `www.homepage.it` and click to the URL within the page
- Intercept the traffic using Wireshark and analyze the behaviour of sslstrip

3.2.1 Countermeasure

HTTP Strict Transport Security (HSTS) is a web security policy to protect against protocol downgraded attacks. Declaring the HSTS policy, the web server forces a browser to use HTTPS. This policy is communicated by the server to the user agent via an HTTPS response header field named *Strict-Transport-Security*.

3.3 Exercise 3 - HSTS Bypassing

The HSTS policy is associated with a specific domain name and changing just by one letter the domain name, the browser will not apply the policy anymore. For instance, an 'l' (uncapital L) could become an 'I' (capital i). The diagram in figure 9 better shows the behaviour of this technique:

1. The user agent makes a request for a non-secure webpage
2. All the traffic reaches the attacker's virtual machine
3. Using iptables the attacker redirects all the HTTP traffic from the port 80 to the port where sslstrip is running
4. The HTTP traffic is forwarded to the server and the response is sent to sslstrip
5. Sslstrip downgrades each `https://` link into an `http://` one, moreover it modify the domain name to something similar
6. The stripped webpage is returned to the user agent
7. When the user agent requests the stripped link a DNS requests is made

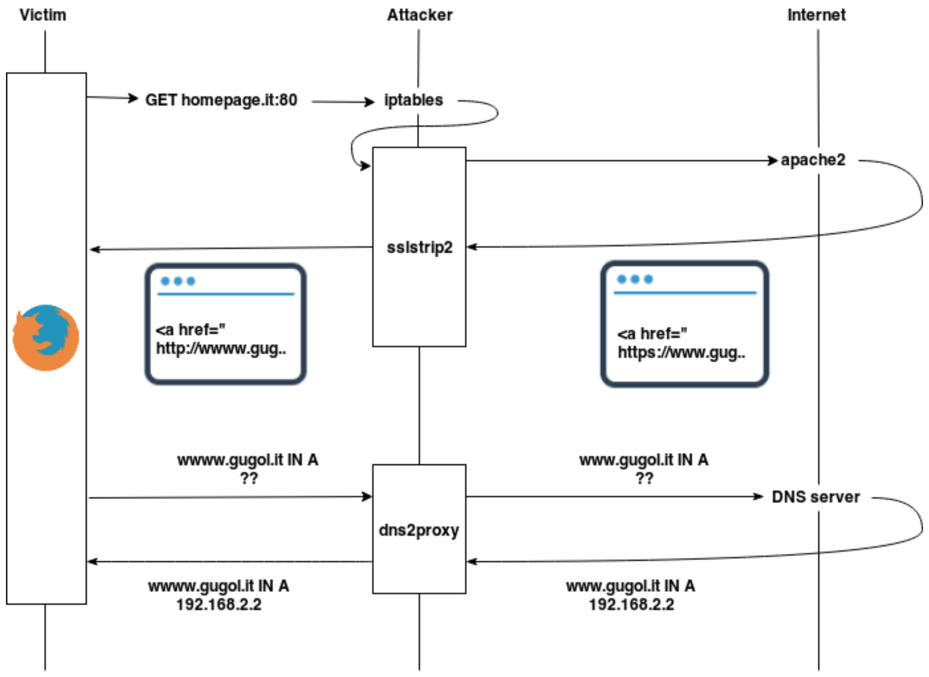

Figure 9: Diagram of the HSTS bypass attack

8. The DNS request passes through dns2proxy and it is manipulated in order to be solved by the DNS server
9. The legit response is received by dns2proxy, altered again and finally returned to the user
10. The user navigates to the webpage with a slightly different domain using HTTP protocol

To correctly mount this attack, you have to follow this procedure:

- Mount a MitM attack
- Implement the missing code in `sslstrip/URLMonitor.py` and execute it `./sslstrip.py -l <sslstrip_port>`
- Using iptables, create a specific rule to redirect the traffic from the port 80 to the port where sslstrip is running
  ```
  sudo iptables -t nat -A PREROUTING -p tcp --destination-port 80
  -j REDIRECT --to-port <sslstrip_port>
  ```
- Implement the missing code in `dns2proxy.py` and execute it `./dns2proxy.py`
- Create a specific rule to redirect all the traffic in the attacker virtual machine changing the destination ip.
  ```
  sudo iptables -t nat -A PREROUTING -p udp --dport 53 -i enp0s8
  -j DNAT --to <attacker_ip>
  ```
- Analyze the behaviour using Wireshark, being sure that all this trick has being working properly

3.3.1 Countermeasures

Unfortunately, there is no default technique to prevent this type of attack. The user must always check the correctness of the URL in the address bar.

3.3.2 Solution of the exercise

Missing code in `sslstrip/URLMonitor.py`

```
if host[:4] == "www.":
    fake_domain = "w" + host
else:
    fake_domain = "web" + host
```

Missing code in `dns2proxy.py`

```
if host[:5] == 'wwww.':
    real_domain = host[1:]
elif host[:3] == 'web':
    real_domain = host[3:]
```