ARP Spoofing and its exploitations

Introduction:

In this paper I will present ARP spoofing and how can we use it to attack a victim and how can we protect of this attack. When you read this work, you must keep in mind that in real life is little bit complicate to use this attack, because you need access at victim's LAN (Local Area Network) network and this can be difficult if a victim uses a strong password for his router Wi-Fi   that we cannot break with a brute force attack. This attack can be performed in real life when a victim uses a public LAN (Local Area Network) such as University Wi-Fi.

ARP Spoofing:

Short Review:

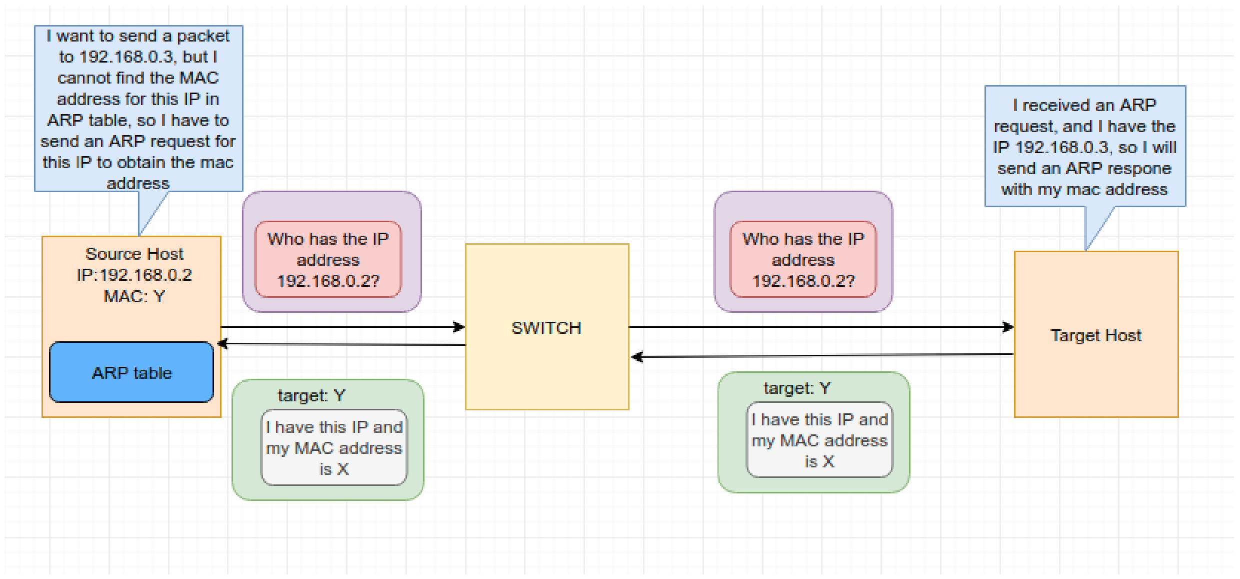
Nowadays, although ARP Spoofing attack has become extremely popular and it was created a lot of tools for its exploitation, more computers and routers ignore it and they do not try to use something to protect against it. Therefore, it is a good to know attack because it can still be used.

ARP Protocol description:

In first, to understand ARP Spoofing you must know "What is ARP?", "Why do we use it?" and "How does it work?."

The ARP (Address Resolution Protocol) is a communication protocol defined in 1982 by RFC 826. It is used to associated link address with internet layer address such as IP address with MAC address. It is a critical protocol because without it we do not have communication over the internet. This protocol is used only in LAN network to facilitate communication between two hosts, because in a single network for communication we use MAC addresses, no IP addresses. Therefore, if we want to send a network package to IP x for example, firstly we must know what is the MAC address for this IP.

In the bellow picture is presented how this protocol work in real life when a person wants to send a network package to another.

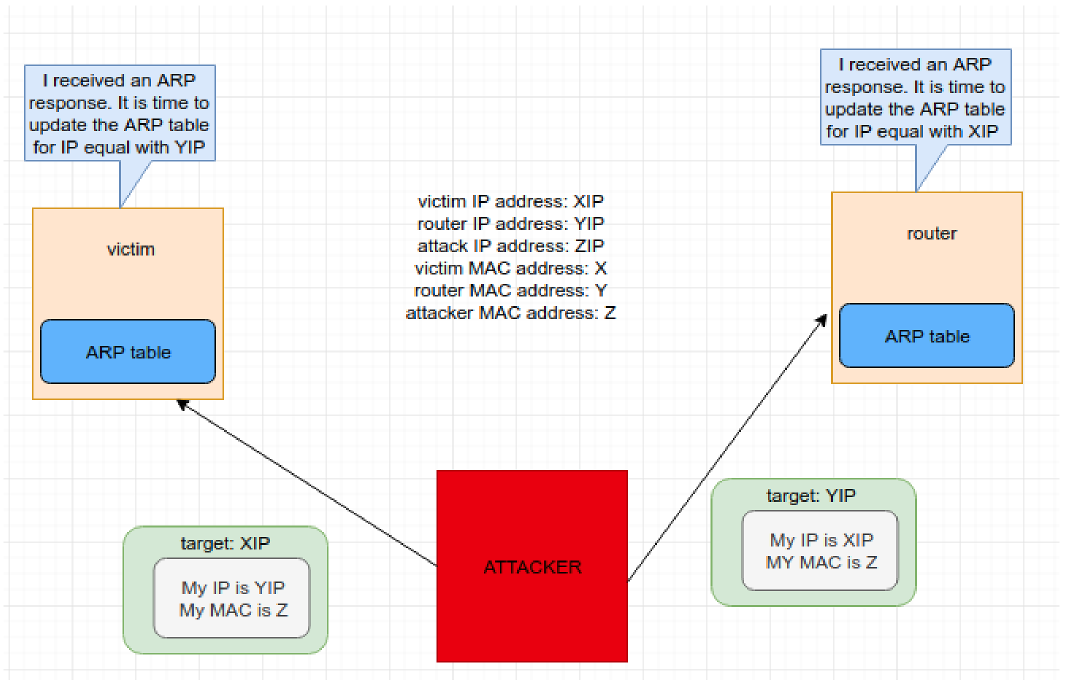

I want to send a packet to 192.168.0.3, but I cannot find the MAC address for this IP in ARP table, so I have to send an ARP request for this IP to obtain the mac address
Source Host IP:192.168.0.2 MAC: Y
ARP table
Who has the IP address 192.168.0.2?
target: Y
I have this IP and my MAC address is X
SWITCH
Who has the IP address 192.168.0.2?
target: Y
I have this IP and my MAC address is X
I received an ARP request, and I have the IP 192.168.0.3, so I will send an ARP respone with my mac address
Target Host

From previous image we can observe that if a user wants to send a package to IP x, then he will send the package at target with MAC equal with MAC address took from ARP table if it exists.

ARP Spoofing description:

ARP Spoofing use a big vulnerability from previous protocol, where Source Host override ARP table with MAC address contained in ARP response without additional validation such as “Does this ARP response correspond to an ARP request packet?.”

Therefore, an attacker can tell the router that he is the victim and the victim that he is the router using the ARP response. After process described in the bellow image, the attack is completed and every packet send from victim to router or from router to victim will pass the attacker first and he can modify it.



ARP Spoofing implementation:

For implementation I used python and scapy library. The entire code can be found at https://github.com/iulius98/hack-tool. If you want to use it you have to install dependencies from file “spoofer_dependencies” before, after that you can view more details about parameters with command “python3 arp_spoofer -h”. This code can be run only on the Linux machine.

So far ARP spoofing does not do something special. We can only view the data from the packets that are not encrypted, but we cannot do a deep attack now. To be able to do more things, such as code injection we need to modify some packets. This we will study this in continue.

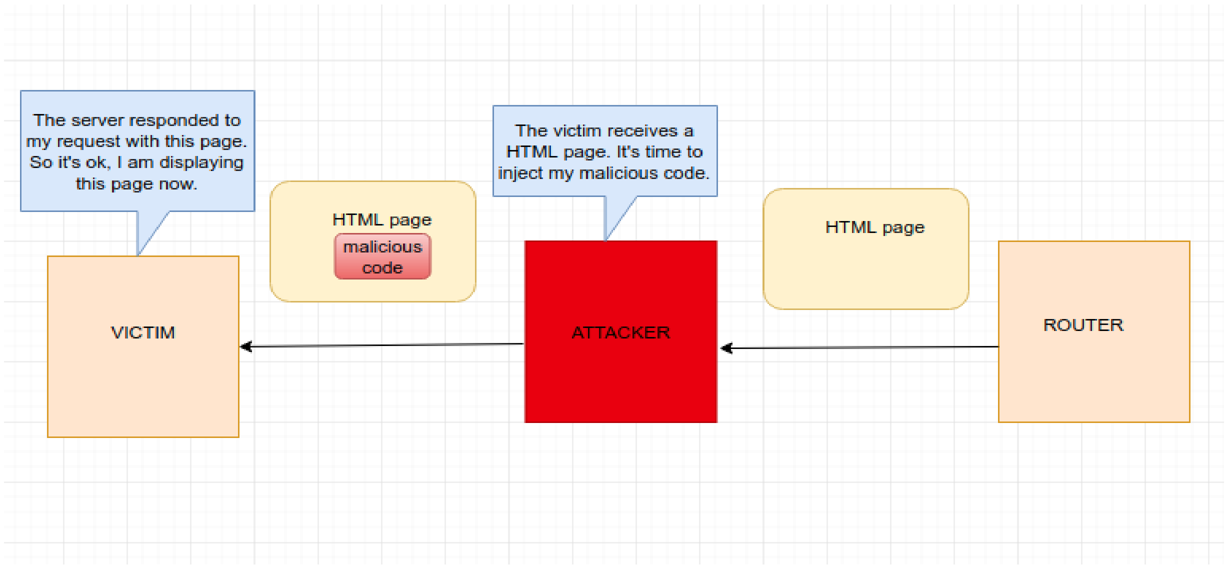
Code Injection:

Short Review:

This attack is based on ARP spoofing, because we must be in the middle of the connection between router and victim. The purpose of this attack is to run malicious code on the victim machine. The result of this attack can be disastrous because attacker can download or running every script that he wants on victim machine. For example, if we inject code into streaming web site such as Netflix we can run a brute force attack on the victim machine, because the victim will spend enough time on that browser. Another example, if we inject code into a bank website, when a victim saves his account in cookies, we can do additional transactions instead of him. Therefore, when you can inject code in the victim's machine the limit is the sky.

Attack Description:

We can suppose now that we are in the middle of the connection. The idea in this attack is quite simple, we modify the packets from the server (in other words from the router) to the user that contain HTML code and inject my javascript malicious code. The following image describes this idea.


The server responded to my request with this page. So it's ok, I am displaying this page now.
HTML page
malicious code
The victim receives a HTML page. It's time to inject my malicious code.
HTML page
VICTIM
ATTACKER
ROUTER

In the picture above things seems to be easy, but they are not, because it is little bit complicated to modify packages in flight. To be capable to modify a package in the flight we must pass some obstacles.

The first problem is about content load compression, because we are not capable to inject the code in load if we do not understand the content. Some browsers and servers support a compression of text because they want to optimize the data which they transfer over the internet. To pass this obstacle we must tell to server that the client does no support data compression. For this when we intercept a message from the client that contains the header "Accept-Encoding: types accepted "we have to remove it from the package to understand the server that the client does not support data compression. For this we can use the following regex expression "Accept-Encoding:*?\\r\\n".

The second problem is about "checksum" and "len" of the packet, if these are not ok the browser will decline the server response. After injection we have to recalculate them. Fortunately, the scapy library does everything for us if we remove these fields from the packet.

The third problem is about the length of the load, because sometimes data are too bigger for a single packet, we have to send all in more packets, and the first packet must contain Content-

Length to know the browser when the message is sent. Therefore, if we inject the code, the content of the message is modified and we have to recalculate Content-Length. To solve this problem, we intercept the first message that contain the Content-Length and replace it with Content-Length + (length of the malicious code).

The fourth problem is about HTTP version. HTTP/1.1 does not use Content-Length and it transfer data in chunked of the same size. Therefore, we cannot inject additional code in these packets. A solution for this problem is to modify the first HTTP request where the client-supported version of http is specified. When server see that the client support at most HTTP/1.0 then it will send Content-Length in first packet response, if the content is too big.

The fifth and most difficult problem to solve is about https. We have to downgrade https to http to be able to read the message. This can be done using sslstrip. In this paper I did not spend so much time on sslstrip, but the principal idea is to create another server on the attacker machine as a proxy. This proxy use https protocol to communicate with target server and when the data are sent from the target server to client it will use only http. In this way the target server does not know that it does not communicate with the client without https and client does not know that server support https.

Code injection implementation:

The code can be found in the previous GitHub repository in the file code_injector.py. For this injector we need additional library "netfilterqueue" that allow us to process the packets that passing through the attacker's machine. As in previous tool we can run this app only on Linux machine because we use some specific command line tools that are only available on Linux.

Testing attack:

Steps:

1.Download Code.

2. Install python3 and its libaries from the dependencies file (github)

3. Start Arp Spoofing from attacker machine (kali machine)

4. Start code_injection attack from attacker machine (kali machine)

5. Access from the victim machine link http://info.cern.ch/

6. You have to see an alert message on your screen.

Protection:

It is little bit difficult to create a software on your machine that protect against Arp Spoofing. Now we can detect it and block the traffic from the attacker machine, but this will block all the host traffic until the attacker stops. An example of this application can be seen in previous GitHub repository(protector.py) where I analyzed every ARP response and verified if the MAC address is good if not, I blocked the host that sent that malicious packet using windows firewall. This can be a solution, but is not enough because it has a negative impact on the user. The best solution for local users is to use a VPN, because it will encrypt the traffic and attacker cannot decrypt it because the user has pre shared secret key with VPN server. The companies have more solutions for this attack such as Switch Security, Static ARP Tables etc.

Conclusion:

In conclusion, Arp Spoofing is an attack that most people ignore it, but any of us can be a victim for this attack. Everyone who has a smartphone uses public Wi-Fi networks from common spaces. Anytime on those networks can be an attacker that want to steal some money from the victims. In this paper I presented only code injection attack, but there are many others attacks that can be exploited such as DNS spoofing, file injector etc. Following this work, you can look for a more efficient way to block ARP spoofing using software application without affecting the user.