DoSing a router

Abstract:

As technology advances, the number of vulnerabilities increases as well. DoS attacks can be performed on various pieces of equipment, or levels of the OSI model. A typical example of DoS attack, TCP SYN flood packet, performs on the transport layer, the DoS carried out in this project performs on the data link layer, specifically it will be about deauthentication since it'll theoretically target a wireless network, by DoSing a router. The strategy behind DoS attacks, however, is usually the same: Figure out a move to get the target's attention and flood it with that same move. Aircrack-ng is a software suite fork of Aircrack, made by Thomas D'Otreppe, it provides different tools to carry out several wireless exploitations. It requires a wireless interface which allows for monitor mode. This project will enlist, explain, and show, if possible, all the tools and shell commands to carry on a DoS attack on a router, using Aircrack-ng.

Motivation:

This project would aim at pushing the reader to learn more about DoSing in general and has the intent of providing a detailed network DoSing tutorial, while assuming the reader has minimal cybersecurity and network skills.

Tentative timeline:

Install proper wi-fi adapter's driver. Ensure Kali Linux VM OS is up to date and compatible with wi-fi adapter's driver; test and record data from testing on my router. Develop an initial paper by 11/11. After that focus on enlisting the shell commands and explain what each one of them does in the paper by 12/1/2024.

Group members: Alessandro Vivaldi

Literature review:

Before discussing the method, which will be used to DoS a router, here's a brief explanation, synthesized from "Resisting SYN flood DoS attacks with a SYN cache" by Lemon Jonathan, of how SYN flood DoS attacks work. First, this type of attack operates at the transport layer because it exploits the way Transfer Control Protocols (TCP) work. TCP requires a connection to be established before two devices, a client and a server for example, can communicate. The way TCP establishes a connection is known as three – way handshake. Client sends a synchronize (SYN) packet to a server, which replies with a synchronize – acknowledged (SYN-ACK) packet, and the client then replies with and ACK packet. Upon the first SYN the connection status is known as initiated, on the client's side, and incomplete on the server's side; the connection status is known as complete on the client's side upon receiving the SYN - ACK packet, and ultimately the connection status becomes complete as well on the server's side, once the final ACK packet is received from the client.

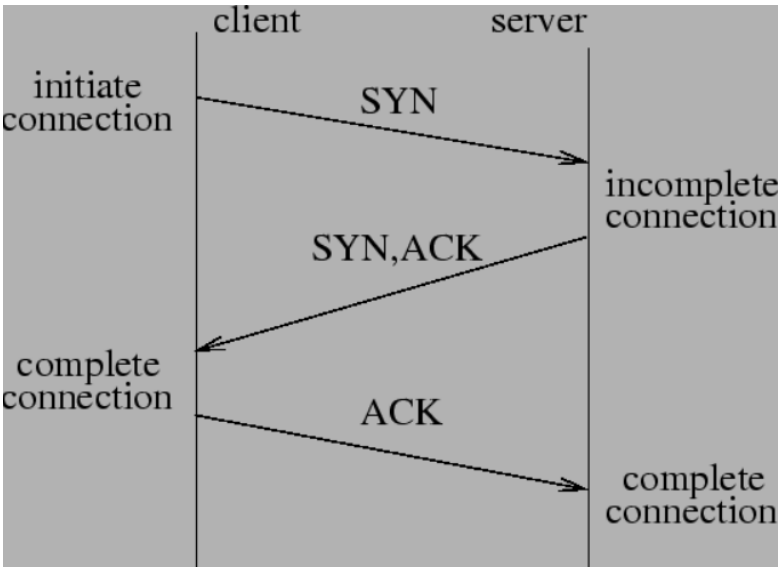


Figure 1 - Three - way Handshake; from Resisting SYN flood DoS attacks with a SYN cache, Lemon J.

The attacker will keep sending SYN requests, creating incomplete connections, which the server will try to respond to. These responses (SYN-ACKs) though will be sent to spoofed (fake) IP addresses, which won't ACK in return, leaving the connections incomplete, allowing the attacker to create more incomplete connections than the server can respond to, exhausting its resources, and ultimately denying services to legitimate users.

Methodology:

Tools required for this project include: a computer, VirtualBox, linux OS virtual machine, kali already has Airmon-ng installed; a Wi-Fi network adapter which supports monitor mode and an internet Wi-Fi connection. The DoS was conducted on a small network, which only used one router. The environment was safe, and no user was affected during the DoSing. The OS of the host machine used for this paper is Windows.

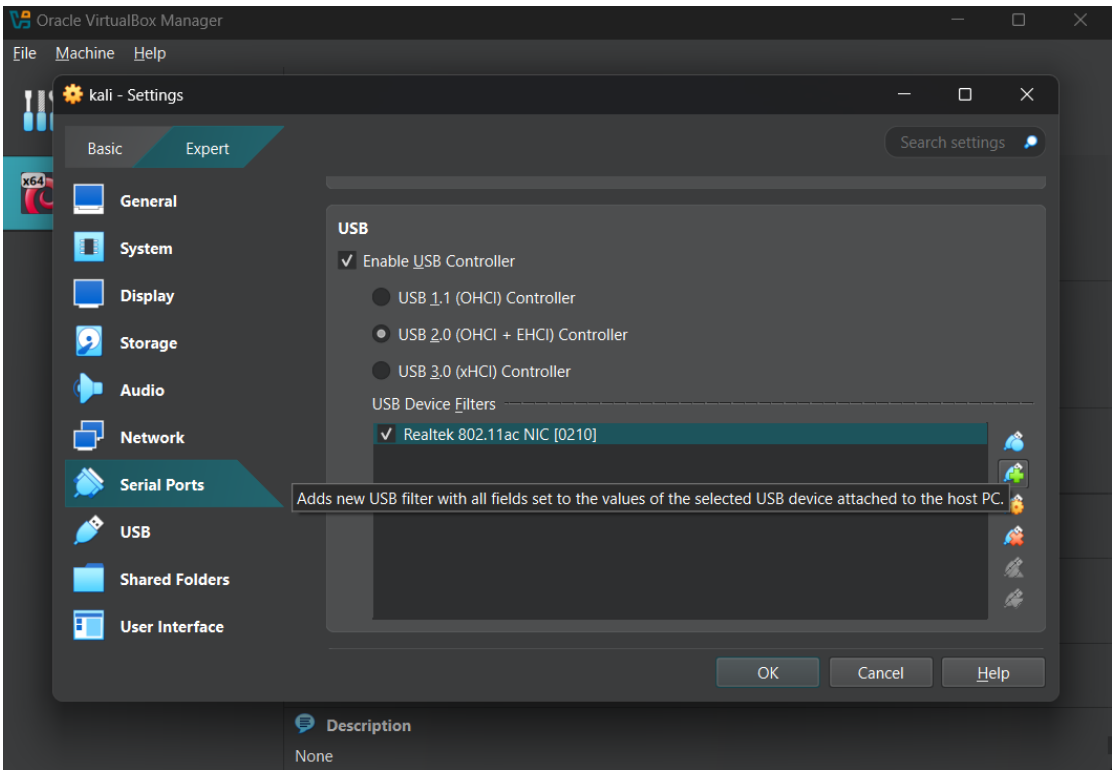
Airmon – ng is a tool used to for wireless network auditing. It can however be exploited to perform a DoS attack. DoS attacks carried through Airmon – ng, are known as deauthentication attacks. These attacks focus on disrupting connections at the link layer; in poor words these attacks overwhelm routers, also known as Access Points (AP), by exploiting the Wi – Fi IEEE 802.11 protocol. This protocol includes different frames for various purposes, including deauthentication frames. These deatuthentication frames are usually sent by the AP to the client to disconnect the client, or by the client to the AP for the same purposes. In this kind of attack, the attacker continuously sends deauthentication packets with a spoofed MAC address to the AP. The spoofed MAC addresses may or may not match the MAC addresses of legitimate users connected to the AP. The repeated transmission of these frames, overwhelms the AP, ultimately leading it to deny internet connection to legitimate users. Since the attack discussed in this paper is a DoS attack that targets the whole network, the spoofed MAC addresses won't need to match any legitimate user's MAC address.

Devices, send and receive signals on specific ranges of frequencies to communicate. These ranges are known as bands. Routers come in many ways depending on various attributes, such as bands. Some routers may have only one band, others multiple. When DoSing a router it's important to know what type and how many bands it uses. Dual-band routers are very diffused, the one used for this project is a single-band router, which transmits on 2.4 GHz.

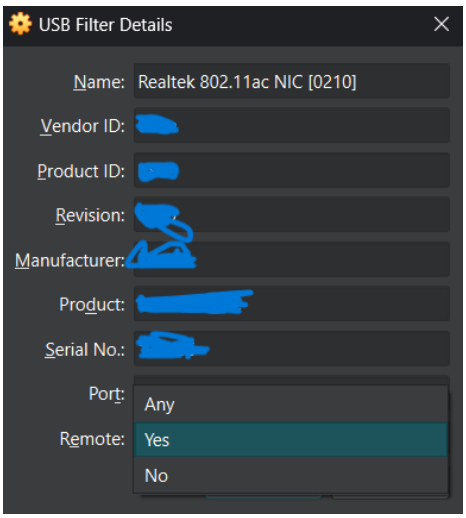
Virtually every computer has a network adapter nowadays. Network adapters can provide many different modes, depending on the needs of the user/s. Monitor mode, however, isn't a very common mode to have on a network adapter, or at least, it isn't provided by network adapters which are used for common reasons. A network adapter in Monitor mode acts as a sniffer, or packet listener, meaning, it allows users to monitor network traffic within range, without connecting to any network. The choice of network adapter should be made considering two aspects; first, the OS kernel version must be supported by the adapter's chipset; secondly, the adapter's driver. Although these adapters can be plugged into a usb port, they still need a driver to function. The driver can be a very painful thing to search, sure there are open-source drivers however not everyone coincides with the OS kernel version of the OS, it's usually better to consider these factors before purchasing the network adapter.

Discussion:

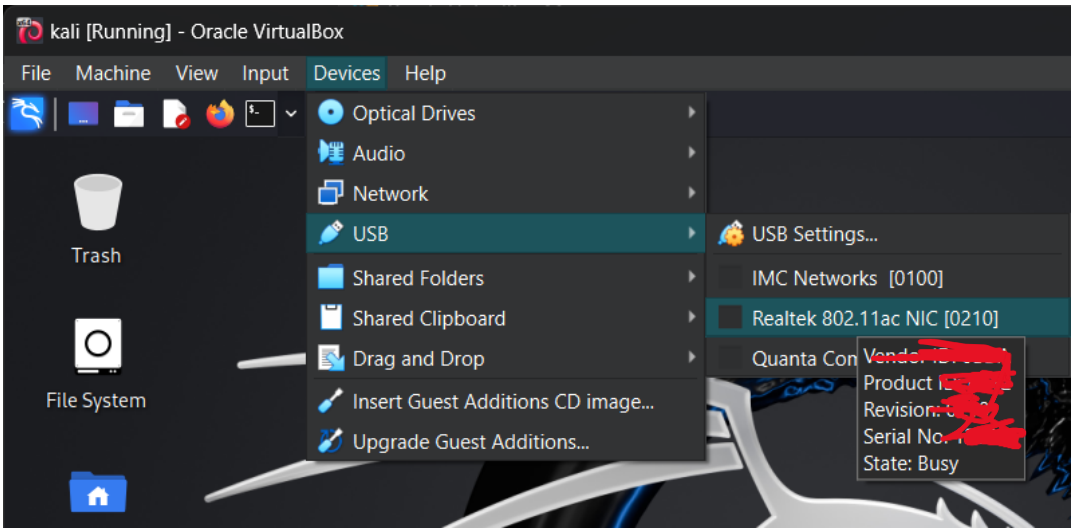
Trying to set up the adapter for the Virtual machine (VM) to detect it may give some trouble at first. In order to avoid this, (after plugging it in the computer) the device has to be enabled, through the device manager (on Windows). After that, add the device filter to the VM, under: machine/Settings/USB/Add.



The filter has to be edited, to allow remote option.



Once the VM is running, the adapter must be checked in the usb settings.



Once the driver is installed the network adapter can be found iwconfig.

Iwconfig shows the network adapters recognized by the machine even if one is not turned on. Ifconfig shows the adapters only if they're on. The network adapter's name is wlan0;

```
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        [redacted]  prefixlen 64  scopeid 0x0<global>
        [redacted]  prefixlen 64  scopeid 0x0<global>
        [redacted]  prefixlen 64  scopeid 0x20<link>
        [redacted]  txqueuelen 1000  (Ethernet)
        RX packets 17  bytes 9181 (8.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 51  bytes 11246 (10.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        [redacted]
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 480 (480.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 480 (480.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        [redacted]  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The next steps involve, shutting down the network adapter, killing every process which may interfere with the adapter, switching the adapter's mode to monitor, and turning the adapter back on.

"sudo ifconfig wlan0 down" shuts the adapter down; "sudo airmon-ng check kill" takes care of the interfering processes.

```
┌──([redacted])-[~]
└─$ sudo ifconfig wlan0 down
[sudo] password for abwa:

┌──([redacted])-[~]
└─$ sudo airmon-ng check kill

Killing these processes:

    PID Name
   2296 wpa_supplicant
```

"airmon-ng start wlan0" turns the adapter back on in monitor mode.

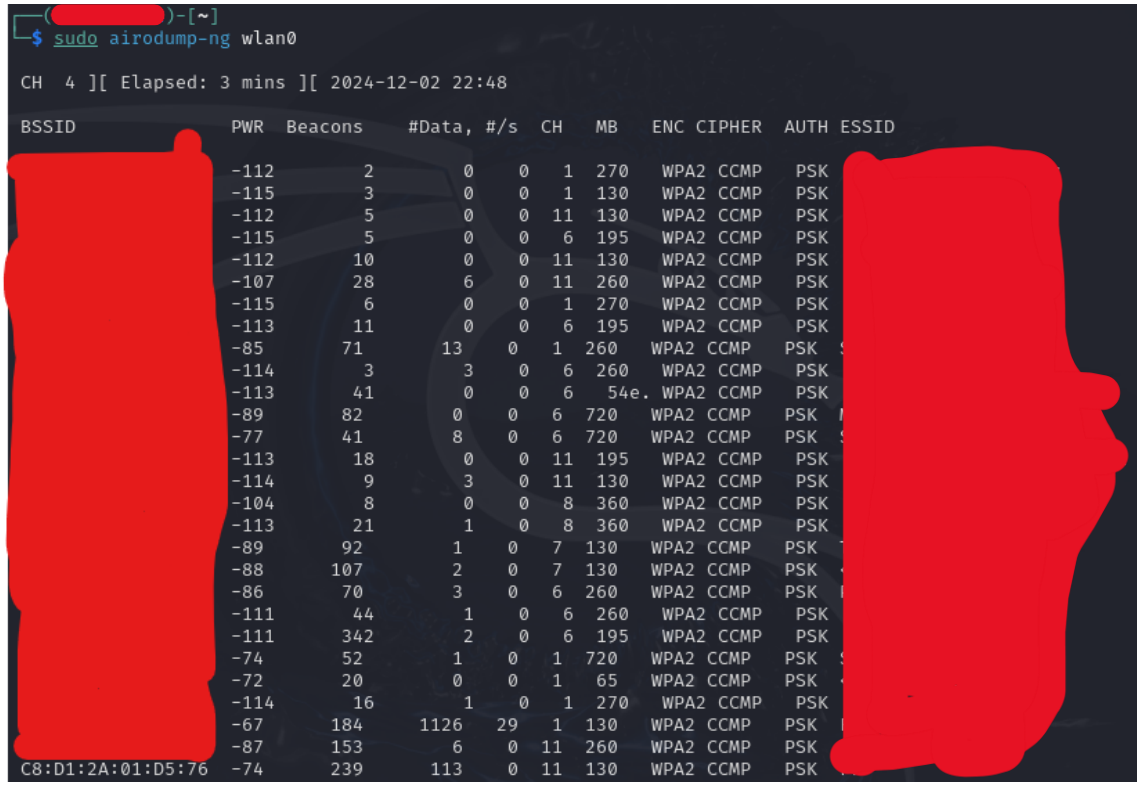
```
┌──([redacted])-[~]
└─$ sudo airmon-ng start wlan0


PHY     Interface       Driver          Chipset

phy0    wlan0           rtl88×2bu       Realtek Semiconductor Corp. RTL88×2bu [AC1200 Techkey]
                (monitor mode enabled)
```

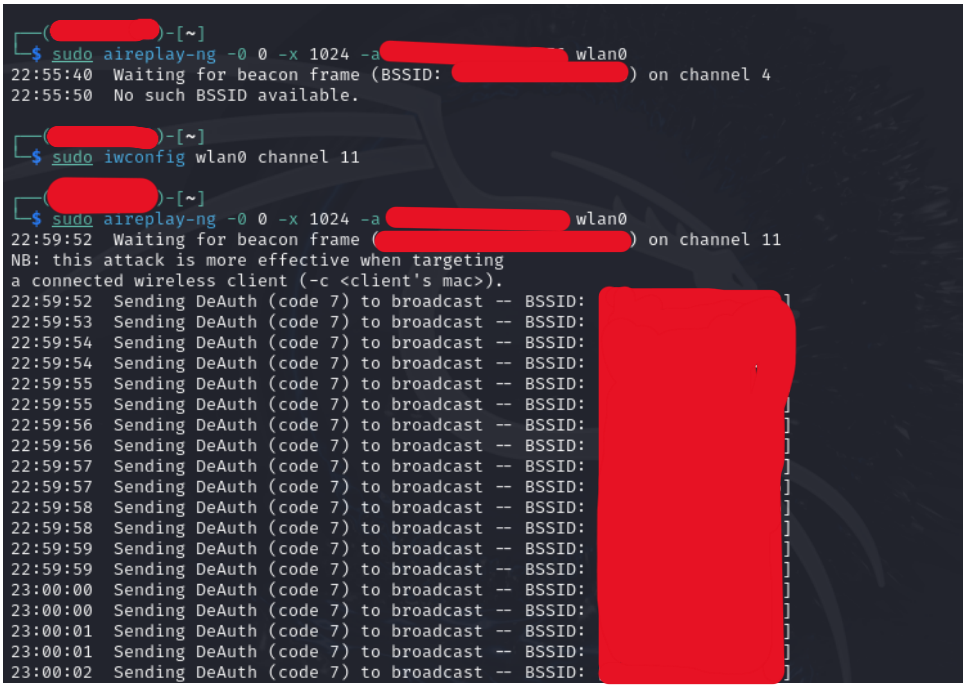
Everything is now set up, for the user to monitor the surroundings and find the MAC address of the access point to DoS.

Run “sudo airodump-ng wlan0” to detect access points.



The targer AP is F7, the name can be seen on the last line of the last column, the ESSID field.

The BSSID field contains the MAC address of the AP. The CH field indicates the router is on channel 11. This means that the Network adapter needs to be put on channel 11 as well otherwise it won’t work.



Conclusion:

This project demonstrated how network deauthentication-based DoS attacks can effectively disrupt wireless networks by overwhelming routers with spoofed packets. Through the use of Aircrack-ng tools and a properly configured wireless network adapter, it was possible to simulate the methodology of such attacks, illustrating vulnerabilities inherent in the IEEE 802.11 protocol.

The experiment also highlights the need for stronger security mechanisms in network protocols to mitigate such exploits. While this project was conducted in a controlled and safe environment, it shows how easily malicious actors could use these tools to target critical infrastructure.

This study also serves as a foundational guide for those new to cybersecurity, illustrating the tools and processes involved in network penetration testing. By understanding the vulnerabilities at play, we can better equip ourselves to build more robust and resilient networks.

REFERENCES:

Lemon, J. (2001, April 12). *Resisting syn flood dos attacks with a syn cache*. USENIX. https://www.usenix.org/legacy/event/bsdcon02/full_papers/lemon/lemon_html/

Aircrack website. https://www.aircrack-ng.org/doku.php?id=tutorial