**MAC** **address** - media access control – is a permanent identification sequence. This sequence can be changed. Devices on a network use the MAC address to communicate.

Changing MAC address improves anonymity. It also allows you to impersonate other devices and bypass filters.

**Interfaces** – Eth0, Wlan0, Lo – these interfaces are all network interfaces. Eth0 is the Ethernet network. Wlan0 is the Wifi network and Lo is the localhost network.

**Terms** – INET refers to the IP address. ETHER refers to the MAC address.

**Commands**

|  |  |
| --- | --- |
| sudo ifconfig | Lists all interfaces and details. |
| sudo ifconfig <interface> | Lists all details for the interface |
| sudo ifconfig <interface> (down|up) | Disconnect/Connect interface. |
| sudo ifconfig <interface> hw ether <new mac> | Spoof interface mac address (a.k.a. ether) |
| netdiscover -r <ip or range> | Find connected devices on same network. |
| arp -a | List ARP tables for the machine. |
| arpspoof -i <interface> -t <target ip> <new ip> | Forces target on the interface into updating its ARP tables to point our MAC address to the new ip (usually the router). |
| echo 1 > /proc/sys/net/ipv4/ip\_forward  **OR**  sysctl -w net.ipv4.ip\_forward=1 | Write 1 (true) on the ip\_forward file which enables packet forwarding through the machine. |
| route -n | Lists the gateways in the network connected. This is helpful to find out the router’s IP address |
| iptables -L (--list) | Check the existing iptables rules. |
| iptables -I FORWARD -j NFQUEUE --queue-num 0 | Creates a new rule on the FORWARD chain of the type NetFilterQUEUE on the machine and give it the identity number of 0. This queue can then be accessed by python to manipulate all packets. |
| iptables --flush | Clear iptables. Leave only defaults. |

**Spoof MAC address**

sudo ifconfig eth0 down

sudo ifconfig etho0 hw ether <new mac>

**Network scanner**

Virtual Machines (VMs) cannot access the host machine wireless card due to the concept of virtualization. To be able to scan wlan (a.k.a. wi-fi) networks, you must use a wi-fi USB card such as Alpha wi-fi cards. The USB ports must be enabled on the VM machine to be able to use the host machine’s USB ports.

**Nat Network**

If there is no wi-fi card available, then to test the concepts you must have 2 VMs running. One will be the target, and another will be Kali VM. The connection between the two should be done via a Nat Network. To set up one go to (VirtualBox) Files > Preferences > Network > (click + sign). [YT video.](https://www.youtube.com/watch?v=y0PMFg-oAEs)

To discover devices connected to the same network:

* Connect using the right interface (wlan0, eth0, etc.)
* Use netdiscover -r <ip>/<range> to find connected devices
* It may be required to disable NAT network.
  + (VirtualBox) Devices > Network > uncheck "connect network adapter"

**ARP - Address resolution protocol**

The ARP links machine’s IP address to MAC addresses which are then used for communications. Basically, when an ARP request is sent, a broadcast message packet reaches everyone on the network and asks, *"who has <some ip>? Tell <sender ip>"*. Only the target machine, whose IP is <some ip>, will reply with its <mac> address. Finally, the initial sender updates its ARP tables which can be seen by running arp -a.

**A few notes about the tutorial:**

The tutorial seems to be using scapy 2.4.2 yet 2.4.3 is out. There is a namespace conflict in PyCharm so importing scapy.all may cause an issue. If you see the squiggly line when importing scapy.all as scapy, simply ignore it. Sadly, this may also prevent you from using autocomplete and intelisense.

**Issue #1 – “text.kerning\_factor” message:**

Commenting out this line fixed the message that kept coming up about "text.kerning\_factor" [Link to SO](https://stackoverflow.com/questions/61171307/jupyter-notebook-shows-error-message-for-matplotlib-bad-key-text-kerning-factor)

**ARP Spoofer**

airspoof -i <interface> -t <target ip> <new ip>

Before running this command, it may be required to enable port forwarding on the proxy machine.

echo 1 > /proc/sys/net/ipv4/ip\_forward

This needs to be run with sudo but it may be required to change to sudo user with sudo su.

ARP spoofing only works if the attacker continuously sends ARP packets to spoof both the router and the target machine. It is possible, however, that newer routers can identify such attacks and block the internet connection on the target machine thus rendering the attack useless.

However, it does seem to work if the right workflow is achieved. In this order:

* Start spoofing attack with timer of 3 seconds.
* Start packet sniffer
* Enable packet forwarding

**Scapy classes info**

Using scapy.ls(<class>) will list all the fields that can be set.

**Issue #2 - NetfilterQueue**

Does not work with python 3.7 or 3.8. Works with python 2.7.18

On PyCharm, simply add an interpreter for Python 2.7. When running use python instead of python3.

It may be required to run the following to install the package. This fixed the issue above.

sudo apt-get install python-netfilterqueue

**iptables**

[More info here](https://www.howtogeek.com/177621/the-beginners-guide-to-iptables-the-linux-firewall/)

Essentially this is a firewall utility that checks whether connections to or from your machine can happen based on the rules defined.

There are 3 chains:

**INPUT** – Controls the behaviour of incoming connections. Ex: a user tries to SSH into your machine. Before it happens, there will be an attempt to match the IP and port to the rule in the INPUT chain.

**FORWARD** – This chain is for routing packets through your machine. Like a router, data is always coming in but it’s rarely destined for the router itself. This allows you to set up your machine as a pass-through device so you to capture all packets.

**OUTPUT** – Same as INPUT but concerned with outgoing connections.

Connections can be:

**ACCEPT** – Accept connection.

**DROP** – Drop connection. No error sent. Acts like connection never happened.

**REJECT** – Reject and send error.

**Creating a proxy**

In order to perform a MITM attack, a proxy need to be created to divert the packets through our machine so that they can later be exploited/manipulated. First a queue needs to be created.

Use the following to create a new iptables rule:

iptables -I FORWARD -j NFQUEUE --queue-num 0

This will allow us to manipulate all packets in the queue.

**Connecting Alfa card**

The only way I got it to work was to go to (VirtualBox) Settings > Network > (select “Not attached”)

Reboot and once Kali boots up, connect the device.

I have also ran previously the following which may have contributed.

sudo apt-get install firmware-atheros

Note – Connection to wifi card seems to be okay if the workflow is followed.

[More info](https://superuser.com/questions/781514/how-to-get-alfa-awus036nha-to-work-with-kali-linux)

**DNS Spoofer**

[ ] Script for Full workflows

[x] Update arp\_spoofer script to handle IndexError on arrays

[ ] Test NetfilterQueue with python 3.5 and 3.6

[ ] Script: get router IP/MAC => Scan network subnet range =>

**Full workflows:**

Each script can be launched individually. Consider spoofing your MAC address before launching any attacks to increase your anonymity within the network.

**To sniff a target’s HTTP traffic:**

1. Launch ARP spoofer
2. Launch Packet Sniffer
3. Enable packet forwarding
   1. Disable after attack is complete.

Ensure iptables are set to default. Run iptables -F to flush. The traffic is captures in the Packet Sniffer window.

**To redirect/manipulate HTTP traffic:**

1. Enable iptables FORWARD rule type NFQUEUE
   1. Disable after attack is complete.
2. Launch ARP spoofer
3. Launch DNS spoofer attack – this will perform the redirect/manipulation of traffic
4. Enable packet forwarding
   1. Disable after attack is complete.

Once the attack is done, make sure to flush the iptables with iptables -F.