

# CHAPTER 3 - ANALYSING & MANAGING NETWORKS

## ▼ [3.1] ANALYZING NETWORKS

### ifconfig

One of the most basic tools for examining and interacting with network interfaces- `ifconfig` [interface configuration]. It can be used to query active network connections.

→ Try typing

**`ifconfig`** in terminal and you would see [a lot more than these, we are discussing some of it here];

#### 1. **`eth0: flags=8763<UP,BROADCAST,RUNNING,MULTICAST>`**

→ **`eth0`**: short for ethernet 0, (Linux starts counting from 0 instead of 1), its the first wired connection, more wired connections would be namely; **`eth1`**, **`eth2`** and so on.

Also, if you would like [I would], here's kinda complete breakdown, might help some curiosity;

- **UP** = The connection is ON [active]

- **BROADCAST** = Can send messages to everyone on the network at once
- **RUNNING** = It's actively working right now
- **MULTICAST** = Can send messages to specific groups of devices

**Translation:** Your main internet connection is turned on, working properly, and ready to send/receive data.

2. *inet addr:192.168.1.131*

→ currently assigned local **IP address**, which is commonly same or similar for each local network (especially micro home computers)

3. *netmask 255.255.255.0*

→ used to determine what part of the IP address is connected to the local network

4. *broadcast 192.168.181.255*

→ Broadcast address is used to send out information to all IPs on the subnet (your local network).

5. *lo: flags=73<UP,LOOPBACK,RUNNING>*

**lo** = This is your computer's "**internal connection**" (loopback interface)

- Think of it as your computer talking to itself
- Like having an internal phone line within your house

**The flags:**

- **UP** = The connection is ON [active]
- **LOOPBACK** = This connection loops back to itself

- **RUNNING** = It's working

**Translation:** Your computer's internal communication system is working fine.

→ The **lo (loopback address)** interface is actually super important, even though it might seem weird that your computer "talks to itself." [ *I did think what's so cool about talking to myself, so I listed some reasons as to why it indeed is cool for computers lmao* ] Here's how;

- Testing a web server on your own computer before putting it online
- Many programs on your computer need to talk to each other
- It's like checking if your phone can make calls by calling yourself
- You build a website and test it at `localhost:8080` etc.

#### 6. *wlano Link encap:EthernetHWaddr 00:co:ca:4r:ff:31*

→ appears only if you have a wireless interface or adapter. It also displays the MAC address of the device (HWaddr)

### **iwconfig**

If you have a wireless adapter you can use the **iwconfig** to gather crucial information on wireless network such as adapter's IP address, MAC address, mode etc.

## ▼ **[3.2] MODIFYING NETWORK INFORMATION**

Being able to change your IP address and other vital network details, is a tangibly useful skill because it helps you in various contexts, for instance, accessing other networks appearing as a trusted device on those networks. Another good example could be, a Denial-of-Service (DOS) attack, you can spoof your IP so that attack appears to come from from another source, thus helping you evade IP capturing in forensic analysis. Forth mentioned tasks are relatively simple and can be carried out using **ifconfig** command.

### ▼ **[3.2.1] CHANGING IP ADDRESS**

To change the IP address, enter ifconfig command, followed by interface (eg. eth0, eth1 etc.) and the new IP address, eg.

→ ***ifconfig eth0 192.168.181.444*** *[ensure you use valid interface if you do not have eth4 and you try to change IP address for it, that wouldn't work]*

Check you IP address and confirm the change by using ifconfig command. Your new new IP address is assigned.



Modern systems often do not entertain the example illustrated above you can temporarily change your IP address like this if the above method doesn't work;

→

***sudo ip addr del 192.168.111.700/24 dev eth0***

Deletes your older IPv4, change the address based on your existing IP address.

check with ifconfig command, your IPv4 address would have been terminated entirely.

Next,

→ ***sudo ip addr add 192.168.1.150/24 dev eth0***

Assigns a new IP address, the one that you created for yourself and mentioned above. These are just temporary changes which would be reset on a reboot, if you wish to make permanent changes a quick google/GPT search could help you more than I could.

### ▼ [3.2.2] CHANGING BROADCAST ADDRESS & NETWORK MASK

To change the Broadcast address and Network Mask of your network interface, enter `ifconfig` command, followed by interface, NEW netmask and broadcast, eg.

→ *`sudo ifconfig eth0 192.168.1.150 netmask 255.255.0.0 broadcast 192.168.255.25`*

Check you netmask and broadcast address, with `ifconfig` and confirm the changes.

### ▼ [3.2.3] SPOOFING MAC ADDRESS

To change the **MAC** address or **HWaddr** [Hardware Address]. MAC address is globally unique and often used as a security measure. Spoofing MAC address is almost trivial and neutralizes security measure. Thus, a very useful technique for bypassing network access controls. Let's illustrate how;

| *`ifconfig eth0 down`*

→ take down [eth0] network interface.

| *`ifconfig eth0 hw ether 00:11:22:33:44:55`*

→ enter interface name [hw for hardware, ether for internet and the new spoofed MAC address.]

| *`ifconfig eth0 up`*

→ Finally, bring the interface back up for the changes to take place.

Check using `ifconfig` command, HWaddr has changed to new IP address.

### ▼ [3.2.4] ASSIGNING NEW IP ADDRESS [DHCP]

Linux has a Dynamic Host Configuration Protocol [DHCP] server that runs a daemon process that runs in the background called **DHCP** or the **dhcp daemon**. The DHCP server assigns IP address to all the systems on the subnet and keeps log files of which IP address is located to which machine at any one time. Thus, making it a great resource for forensic analysis to trace intrusions. Usually, to connect to the internet from a LAN, you must have a DHCP-assigned IP. Let's see how you can retrieve a new IP without having to reboot or shut down your entire system.

→ dhclient eth0

The dhclient command sends a DHCPDISCOVER request from the network interface specified (here, eth0). It then receives an offer (DHCPOFFER) from the DHCP server (192.168.181.131 in this case) and confirms the IP assignment to the DHCP server with a dhcp request.

Depending on the configuration of the DHCP server, the IP address assigned in each case might be different.

Now when you enter ifconfig, you should see that the DHCP server has assigned a new IP address, a new broadcast address, and new netmask to your network interface eth.



It's plausible dhclient isn't installed (mine wasn't) you can do so by:

```
| sudo apt install isc-dhcp-client
```

```
| sudo apt install isc-dhcp-client-ddns
```

Run the above 2 commands in your Kali terminal and its installed.

## ▼ [3.3] MANIPULATING THE DOMAIN NAME SYSTEM

### ▼ [3.3.1] EXAMINING DNS

DNS is a critical component of the internet, and although it's designed to translate domain names to IP addresses, a hacker can use it to garner information on the target.

## → Examining DNS with dig

DNS is the service that translates a domain name like ***alicesdomain.com*** to the appropriate IP address; that way, your system knows how to get to it. Without DNS, we would all have to remember thousands of IP addresses.

One of the most useful commands is- ***dig*** command, it offers a way to gather DNS information about a target domain.

The stored DNS information can be a key piece of early reconnaissance to obtain before attacking. This information could include the IP address of the target's nameserver, the target's email server, and potentially any subdomains and IP addresses. [translated to: its a freaking gold mine.]

### ▼ [3.3.2] CHANGING DNS

#### Temporary DNS Configuration

For quick testing or troubleshooting scenarios, you may need to implement a temporary DNS change that won't persist across system reboots or network reconnections. The classic approach involves direct manipulation of the `/etc/resolv.conf` file—a straightforward method that provides immediate results without affecting your system's underlying network configuration. Open your preferred text editor with administrative privileges and access the file directly:

```
| sudo nano /etc/resolv.conf
```

Within this file, you'll observe existing nameserver entries, typically pointing to your local router or ISP-provided DNS servers. To temporarily override these settings, simply prepend your desired DNS server to the top of the list. For instance, to utilize Cloudflare's public DNS service, add the following line at the beginning:

## nameserver 1.1.1.1

Alternatively, accomplish this modification entirely from the command line using a single elegant command that temporarily preserves your existing configuration while adding the new DNS server:

```
echo "nameserver 1.1.1.1" | sudo tee /etc/resolv.conf.tmp  
&& sudo mv /etc/resolv.conf.tmp /etc/resolv.conf
```

This approach ensures immediate activation of your DNS changes. However, remember that these modifications remain ephemeral—they'll vanish upon the next DHCP lease renewal, network restart, or system reboot, making this method ideal for diagnostic purposes or temporary network troubleshooting sessions.

## Permanent DNS Configuration

For establishing persistent DNS settings that survive system reboots, network changes, and service restarts, you'll need to configure your DNS servers through your system's network management infrastructure rather than directly editing the volatile `/etc/resolv.conf` file. Modern Linux distributions employ sophisticated network management services that maintain configuration persistence across various system states.

On NetworkManager-enabled systems (prevalent in most contemporary desktop environments), achieve permanent DNS configuration through the network management interface. Identify your active network connection and modify its DNS settings using the NetworkManager command-line interface:

```
# First, identify your connection name
```

```
nmcli connection show
```

```
# Then configure DNS servers permanently
```



```
sudo nmcli connection modify "Your-Connection-Name"  
ipv4.dns "1.1.1.1,8.8.8.8"  
sudo nmcli connection modify "Your-Connection-Name"  
ipv4.ignore-auto-dns yes  
sudo nmcli connection up "Your-Connection-Name"
```

For systemd-resolved environments (common in Ubuntu 18.04+ and other modern distributions), establish permanent DNS configuration by creating a persistent configuration file:

```
sudo mkdir -p /etc/systemd/resolved.conf.d  
sudo tee /etc/systemd/resolved.conf.d/dns_servers.conf  
<< EOF  
  
[Resolve]  
  
DNS=1.1.1.1 8.8.8.8  
FallbackDNS=9.9.9.9  
EOF  
sudo systemctl restart systemd-resolved
```

This methodology ensures your DNS preferences remain intact regardless of network state changes, DHCP renewals, or system maintenance operations. The configuration integrates seamlessly with your system's network stack, providing both reliability and performance optimization while maintaining compatibility with modern Linux networking paradigms.

### ▼ [3.3.3] MAPPING YOUR OWN IP

In Linux, there's a special file called the **host** file, it performs domain name-IP address translation/resolution. The hosts file is located at **/etc/hosts**, and kind of as with DNS, you can use it to specify your own IP address–domain name mapping.

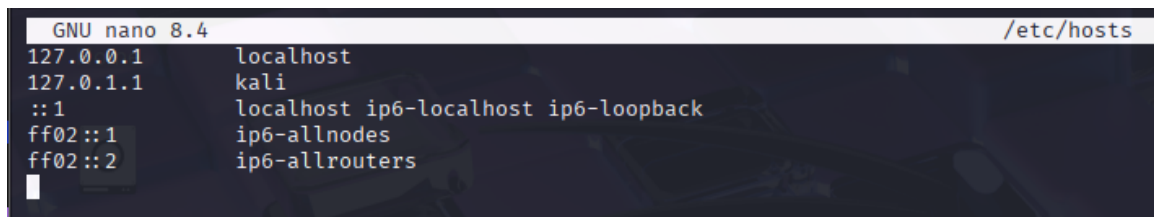
In other words, you can determine which IP address your browser goes to when you enter `www.google.com` (or any other domain) into the browser, rather than letting the DNS server decide.

As distinguished gentlemen and ladies, I'm pretty sure you get the idea here, you should by now.

From the Linux terminal, type in the following command (you can substitute your preferred text editor for nano):

```
| nano /etc/hosts
```

You will be looking at something like:



```
GNU nano 8.4 /etc/hosts
127.0.0.1    localhost
127.0.1.1    kali
::1         localhost ip6-localhost ip6-loopback
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters
```

By default, the hosts file contain only a mapping for your localhost, at 127.0.0.1, and your system's hostname (in this case, Kali, at 127.0.0.1).

As you get more involved in your Cybersecurity and DFIR endeavors, you'd learn about tools like `dnsspoof` and `Ettercap`, you'll be able to use the hosts file to direct any traffic on your LAN that visits [www.microsoft.com](http://www.microsoft.com) for instance, to your web server at 192.168.181.131.