

Network security – basic attacks

Laboratory for the class “Information Systems Security” (01TYM,02KRQ)

Politecnico di Torino – AA 2020/21

Prof. Antonio Lioy

prepared by:

Diana Berbecaru (diana.berbecaru@polito.it)

Ignazio Pedone (ignazio.pedone@polito.it)

v. 3.3 (8/10/2019)

Contents

1	Information gathering	3
1.1	Network Scanning	3
1.2	Port Scanning	4
1.3	Identification of (application) services	5
2	Capture and manipulation of network traffic	5
2.1	Man-in-the-middle	5
2.2	Sniffing attacks	7
3	Additional exercises (optional)	9
3.1	DNS spoofing	9

Purpose of this laboratory

The widespread use of networks and especially of Internet has created on one hand a virtual workspace enabling people to collaborate and communicate in real time, but at the same time it opened up the door to cyberspace attackers. Even if the most sophisticated network security attacks can only be performed by very experienced people with strong motivations and budget, the exercises proposed in this laboratory will prove that also people with basic security notions and only using freely available software can be very dangerous in some contexts.

ATTENTION

Some of the operations described in this document are illegal and are liable to prosecution. The purpose of this document is to present the actions (and tools) just for educational purposes. Students are required to try the attacks only towards the computers located in the laboratory dedicated to the course “Computer system security”, or towards the computers they own. The authors of this document decline any responsibility for actions performed by the participants of this laboratory in violation of this policy.

Software tools

The tools listed below will be used throughout this (practical) laboratory:

Nmap - network and port scanner open source. It is designed to perform quick scanning of networks of big dimensions. For example, it allows to determine which type of operating system a particular network host is running and which (application) services are active at one time. Available for Linux and Windows.

Home page = <https://nmap.org>

Ettercap - open source tool, which allows to perform man-in-the-middle (MITM) attacks and sniffing attacks in a Local Area Network (LAN). Available for Linux e Windows.

Home page = <http://ettercap.github.io/ettercap/>

Wireshark - open source tool (having a user-friendly graphical interface) which allows to capture network traffic. Available for Linux and Windows.

Home page = <https://www.wireshark.org/>

Additional tools

When executing the exercises, you will need also to start, stop or reconfigure some services:

Apache2 - Web server

To start, stop and restart the web server, use the command:

```
systemctl { start | stop | restart } apache2
```

The configuration of server ports (e.g. the listening port) can be found in the file `/etc/apache2/ports.conf`

VSFTP - FTP server

To start, stop and restart the FTP server, use the command:

```
systemctl { start | stop | restart } vsftpd
```

The configuration of server ports can be found in the file `/etc/vsftpd.conf`

For example, to change the `listen_port` to 201, you need to add/modify the line `listen_port=201`

SSH2 - SSH server

To start, stop and restart the SSH server, use the command:

```
systemctl { start | stop | restart } ssh
```

Exim - Mail server

To start, stop and restart the mail server, use the command:

```
systemctl { start | stop | restart } exim4
```

Using the arp commands

You can delete an IP address from the ARP table by using the `arp` command with the `-d` option followed by an address. For example, to delete the IP address 192.168.1.44:

```
arp -d 192.168.1.44
```

If you are not sure which IP address you are looking for then you can look at the ARP table to check the IP against the MAC address by using the `-e` option. For example:

```
arp -e
```

Use of IP

The command `ip` is a powerful command that can be used to analyse and manipulate the routing of IP packets on a Linux machine.

It allows also to remove all the entries of an ARP table (this operation cannot be done in a single step with the `arp` command) with the following command:

```
ip neigh flush all
```

It is useful to add the flag `-s` repeated twice in order to print information about the result of the command:

```
ip -s -s neigh flush all
```

During the laboratory, you will need to execute commands that require `root` privileges. For this reason, we suggest you to log in as the `root` user (pwd: `toor`) from the beginning of the practical exercises.

Network configuration

As you have seen in the previous laboratory for setting up the environment, there is no DHCP server over the VM network. Because of this, you need to configure the networking **before** being able to do the proposed exercises.

Set the IP address with the following command, where *machine_id* is the sum of the number on the name of your VM plus 100 (e.g. for “torsec042” the *machine_id* will be 142, for “torsec107” it will be 207, and so on):

```
ip addr add 192.168.0.machine_id/24 dev eth0
```

1 Information gathering

1.1 Network Scanning

The first phase of the preparation of an attack consists in detecting the target(s) of the attacks. For this purpose, a technique called *Network Scanning* is typically used. The objective of this technique is to obtain information about which hosts in a particular network are active and which ones are not active.

How can this be done in practice? Which is the most common method used to determine whether a certain host is reachable?

→ Ping it! (ICMP Echo Request)

Why (in general) it is not recommended to perform network scanning inside a monitored network?

→ It is easy to detect and trace automatic scanning activity

Once you have identified the active network hosts that are actually running, you can proceed to gain more information about them in order to detect their security weaknesses. In practice, a series of techniques known as *Network Fingerprinting* allow to obtain information on the remote host's operating system. The *Network Fingerprinting* technique is based on the fact that various types of operating systems (e.g. Windows and Linux) implement differently the TCP/IP stack.

The program named `nmap` is a very easy-to-use and powerful network scanner, which allows to detect the operating system running on a remote host, by means of network fingerprinting. For a description of the overall program, use the following command:

```
man nmap + /usr/share/nmap
```

Exercise 1. Two students form a group of two VMs named Alice and Bob (one for the victim and one for the attacker), then proceed as follows:

1. Alice (victim): starts the Apache web server.

-s=scan

-T=TCP

2. Bob (attacker): tries to establish a TCP connection (-sT) on the port 80 (-p 80) of the target host Alice (*IPaddress_Alice*), in order to obtain information about the operating system (-O) running on the victim's machine:

```
nmap -sT -p 80 -O -v IPaddress_Alice
```

1.2 Port Scanning

After discovering the target host, the attacker typically tries to find out which (application) services are actually running on that host. The technique known as *Port Scanning* is used for this purpose. To obtain information about which ports of a particular host are open (waiting for incoming connections) and which ones are closed, a *Port Scanning* tool can be used. By using such a tool, it is possible also to determine for each port: the (default) name of the known service (if one exists, like for example `http` service or `ftp` service), the port number, the port's state (open, filtered by a firewall or by a packet filter, unfiltered) and the corresponding protocol. For a description of the various types of port scanning currently supported by Nmap, you can run the corresponding `man` command, i.e. `man nmap`.

Exercise 2. Two students form a group of two VMs named Alice and Bob (for the victim and the attacker), and proceed as follows:

1. Alice (victim): chooses two services among `http`, `ftp`, `ssh` and activate them.

2. Alice checks that only the services she has selected are actually active (check out by using `lsof -i` or `netstat -ltu`)

3. Bob (attacker): makes a TCP port scan (-sT) on the first 1024 ports of the target host Alice (*IPaddress_Alice*):

```
nmap -sT -Pn -p 1-1024 -v IPaddress_Alice
```

Do you manage to discover which services have been activated by the victim (Alice)?

→ YES!

Why is it useful to avoid pinging the host (the option -Pn)?

→ It takes less time, since we already know that he's up, and we will expose ourselves as little as possible

Exercise 3. Now execute the following operations:

1. Alice (victim) chooses again two services among `http`, `ftp`, `ssh`
2. Next, she starts the chosen services on ports different from the default ones (see the introductory notes for instructions on changing the ports).
3. Bob (attacker) performs a new TCP port scanning on the host of her colleague (Alice).

Have you encountered problems? How do you think they can be solved?

→ nmap shows the open ports number but guesses their service based on the standard ports, to verify that we can try to connect manually to each port to see what are the responses

1.3 Identification of (application) services

Exercise 4. To identify the (application) services running on the open ports on your colleague's machine, try to use the option (-sV):

-sV: Probe open ports to determine service/version info

```
nmap -sV -Pn -p 1-1024 -v IPaddress_Alice
```



For a detailed description of the techniques used by Nmap to identify the services use the command:

```
man nmap https://nmap.org/book/man-version-detection.html
```

In your opinion, which techniques and security tools can be used to defend against the security attacks presented above?

→ Don't leave unused services active. Try to block scans using firewall with whitelist of IPs

2 Capture and manipulation of network traffic

2.1 Man-in-the-middle

Etercap is a versatile tool that can be used to execute (besides sniffing, filtering etc) man-in-the-middle attacks in a LAN. In particular, in this exercise, we will concentrate on the "ARP poisoning" attack.

For a description of the parameters of Ettercap tool and of its configuration, you can run the following commands:

```
man ettercap
```



```
man etter.conf
```

Given the type of attack, you can use the command arp to view the content of the ARP cache:

```
man arp
```

Exercise 5. Three students form a group of three VMs (let's say Alice, Bob and Chuck) and proceed as follows:

1. Examine the operation of the network in normal conditions: Alice, Bob e Chuck exchange ping messages and note down the ARP cache of each host in Table 1.
2. Chuck tries to sniff the network traffic with wireshark

3. Chuck launches tt by starting ettercap in the following way (pay attention to the "/" characters):

```
ettercap -Tq -M arp /IPaddr_host_Alice// /IPaddr_host_Bob//
```

The program starts in interactive mode. Chuck can view a brief help, by pressing the button h. While executing the command, check out the network traffic on the three hosts by capturing the packets with wireshark.

4. Alice and Bob exchange again ping messages.

What happened to the ARP caches of Alice, Bob and Chuck? Verify your hypothesis by printing out the content of the ARP caches with the arp command in Table 2.

What kind of network traffic was able to capture Chuck before the attack?

→ He could capture broadcast traffic or traffic exchanged with him



Role	IP Address	MAC Address
Alice	10.0.2.7	08:00:27:13:36:7b
Bob	10.0.2.5	08:00:27:fe:2a:7d
Chuck	10.0.2.15	08:00:27:5c:65:26

Table 1: ^{Alice's} ARP cache(s) content before the attack

Role	IP Address	MAC Address	
Alice	10.0.2.7	08:00:27:13:36:7b	08:00:27:5c:65:26
Bob	10.0.2.5	08:00:27:5c:65:26	08:00:27:fe:2a:7d
Chuck	10.0.2.15	08:00:27:5c:65:26	08:00:27:5c:65:26
		Alice's	Bob's

Table 2: ARP cache(s) content after the attack

Chuck's ARP Cache is normal (he made the attack)

What type of network traffic is able to sniff Chuck after the attack?

→ He can see also the traffic exchanged between Alice and Bob

Can you figure out from the captured network traffic how the attack works?

→ arp req da Chuck a Alice | arp req da Chuck a Bob | PING dall'IP di BOB ma con il MAC di CHUCK ad ALICE | PING dall'IP di ALICE ma con il MAC di CHUCK a BOB | arp reply da Chuck a Alice fingendo di essere Bob | arp reply da Chuck a Bob fingendo di essere Alice (ultime 2 si ripetono)

What happened to the ARP tables of Alice, Bob and Chuck? Check out.

→ Chuck is the Man In The Middle

Take a look at the Figure 1 and try to respond to this question (where ? can be a *hub*, a *switch*, a *router* or even *Internet*): when is it possible to make a man-in-the-middle attack with ARP poisoning and which are its effects?

→ ?=hub or switch, because if it is a router, then the arp packets don't propagate to other physical networks

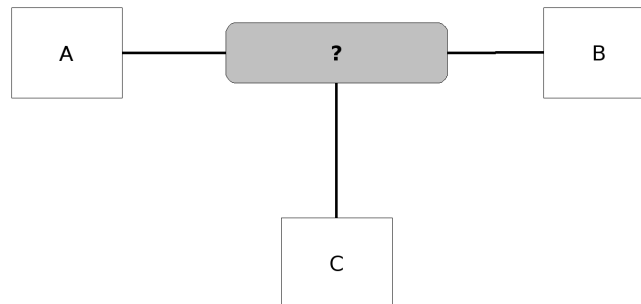


Figure 1: man-in-the-middle

Optionally, try to identify a possible countermeasure to this security attack:

1. Chuck modifies the line `arp_poison_delay` in the file `/etc/ettercap/etter.conf`, by setting it to 1000. Next he restarts `ettercap`.
2. Bob deletes from the ARP cache the line corresponding to the host of Alice. After that, he tries to execute a ping to Alice's host.

What happened? Bob could ping Alice and she could respond without Chuck knowing because Bob deleted his poisoned arp cache, pinged Alice and received her response, before Chuck could respond (poisoning it again). This was possible because the `arp_poison_delay` between 1 arp response and the next was big

What Chuck did at point 1?

What is the meaning of the variable `arp_poison_delay`? Prove again the exercise by setting the variable to 1 or to the default value, 10. This variable controls the poisoning delay after the first 5 poisons. The value is expressed in seconds. You can increase this value (to try

to fool the IDS) up to the timeout of the ARP cache (which depends on the poisoned operating system).

Per protrarre l'attacco è necessario inviare delle ARP reply ogni 10 secondi poiché spesso i sistemi operativi cancellano sistematicamente le voci dell'ARP cache dopo un certo periodo di tempo.

→ Whenever the attack is triggered, since it sends pings from the victims IPs, they are forced to write the poisoned arp cache.

If one victim deletes the poisoned entry and sends a ping before the attacker starts the attack again, he will be able to reach his target without being intercepted, but then his arp cache will be poisoned again

2.2 Sniffing attacks

In this section, we'll experiment with the passive network security attacks. In practice, *sniffing* is a passive attack that consists in capturing the packets passing through our Ethernet network card set in promiscuous mode. In this way all the packets that should be ignored, i.e. the ones that don't correspond to the MAC address of the network card, are instead copied in a buffer.

Actually, you have already tried the sniffing technique in the exercises proposed above, when you have captured the network packets with *wireshark*. In this exercise, we will concentrate on capturing sensitive data (e.g. passwords) exchanged inside application protocols, instead of capturing generic data traffic.


To experiment the sniffing attack, we will use again *ettercap*, which extracts username and password from network traffic exchanged in common services (including FTP). Another useful tool is the *Ngrep* tool (<http://ngrep.sourceforge.net>), which allows to search for pre-defined strings in the captured traffic. In practice, *Ngrep* provides functionality similar to the classical *grep* command, which is applied to the (captured) network traffic.

Exercise 6. Three students form a group of three VMs (Alice, Chuck and Bob):

1. Alice creates the users `alice` and `bob`, and sets accordingly the password:

```
adduser alice
```

```
adduser bob
```

2. Chuck launches the MITM attack with *ettercap* as in the Exercise 2.1
3. Bob using `ftp` command connects to the FTP server running on the machine Alice, he executes login (with his username and the password set above) and then he disconnects 

NOTE

In some cases, the following exercise (option `-e` of *ettercap*) does not function correctly with the virtual machines. The ARP spoofing is performed correctly, but the identification module of *regex* does not manage to recognize the strings.

Try now to extract sensitive information from the mail traffic, by executing the following steps (Alice must have already created users `alice` and `bob` through the `adduser` command as previously explained):

1. Alice configures the `exim` mail server with the command:

```
dpkg-reconfigure exim4-config
```

and by selecting the parameters in the following manner:

- (a) General type of mail configuration: Internet site; mail is sent and received directly using SMTP.
- (b) System mail name: `kali`
- (c) IP-addresses to listen on for incoming SMTP connections: *// leave blank (delete data if present)*
- (d) Other destinations for which mail is accepted: `kali`
- (e) Domains to relay mail for: *// leave blank (delete data if present)*
- (f) Machines to relay mail for: *// leave blank (delete data if present)*
- (g) Keep number of DNS-queries minimal (Dial-on-Demand) ?: No

- (h) Delivery method for local mail: mbox format in /var/mail
- (i) Split configuration into small files ? : No
- (j) Root and postmaster mail recipient: *// leave blank (delete data if present)*

Subsequently starts the server with the command:

```
systemctl start exim4
```

2. Chuck executes the command:

```
ettercap -T -M arp /IP_host_Alice//25 /IP_host_Bob// -e "Credit Card"
```

3. Bob sends an e-mail to alice@kali with the following message "Don't do this in practice: do not send any Credit Card number (like 7865-8993-6282-8282) by mail!" by using an e-mail client named s-nail

```
s-nail -S mta=smtp://IP_host_Alice -S 'from=bob@kali' -s "Exam Security"
alice@kali
```

once you have finished editing the e-mail text message press Enter then Ctrl-D

4. Alice can verify that the user `alice` has received the message with the following command (don't waste time configuring an e-mail client for this exercise)

```
cat /var/mail/alice
```

Try now to spy the web navigation, by executing the following steps:

0. Chuck verifies his X Window System setup

1. Alice activates the service http

2. Chuck enables the access to the display for the user `root`, by opening a terminal and executing the command:

```
xhost +
```

By default, ettercap sets its uid and gid to the user `nobody`. To allow ettercap to open a window in a graphical environment running as root, you have to change the configuration in `/etc/ettercap/etter.conf`.

In this file, Chuck has to set `ec_uid` and `ec_gid` to 0, in order to maintain root privileges.

- 2b. `remote_browser = "firefox -remote openurl(http://%host%url)"`

3. Chuck, starts the browser and launches ettercap with the following command to activate the `remote_browser` plug-in:

```
ettercap -T -M arp /IPaddr_Alice//80 /IPaddr_Bob// -P remote_browser
```

Note: the configuration of the plug-in can be found in `/etc/ettercap/etter.conf`.

4. Now Bob can connect with a browser to the http server running on Alice's machine:

```
http:// IPaddr_Alice/
```

Chuck verifies the web page loaded in his browser.

How can Chuck intercept all the web traffic of Alice?

→

What happened in the previous tests?

How can you defend against these types of sniffing attacks?

→

Try to execute a login by using SSH protocol and to capture the username and password: what happened?

→

3 Additional exercises (optional)

3.1 DNS spoofing

In this exercise, we will see how to apply spoofing technique against DNS.

ATTENTION

It is not possible to do this exercise on the VDiLinux VM because there is no DNS server available over the local network and all VMs are not connected to the internet. Nevertheless, you could try this at home with local VMs

NOTE

In a virtual environment or when the DNS server is in the same network of the attacker and the victim, these tests might not work because the success of DNS spoofing is based on the speed of the response. It depends on whether the attacker manages to send a response corresponding to a DNS request faster than the legitimate server. In virtual environments, we deal with virtual switches (that is software processes that simulate the behaviour of a real switch) which behave both as DHCP and DNS server, so the DNS server of the virtual switch responds faster than the attacker. So, in this case, it might not be possible to obtain a “quicker” response from the attacker.

As said, in this exercise we use `ettercap` for this attack. The attacker will behave as a man in the middle between the victim and the DNS server. Next, `ettercap` will be in charge of filtering the DNS requests originating from the victim and to respond before the DNS server will do so.

Exercise 7. Form a group of two VMs: Alice (the attacker) and Bob (the victim). Next proceed as follows:

1. Alice adds in the file `/etc/ettercap/etter.dns` the command

```
www.polito.it A 130.192.39.120
```

2. Bob controls the content of the file `/etc/resolv.conf`
3. Now, try to understand and check out the functionality of the network under normal conditions: Bob tries to connect to a web site in Internet, for example to the web site www.polito.it. Check out directly the DNS query sent by using the command:

```
host -v www.polito.it
```

You should see information provided by DNS.

→

4. Alice identifies the IP address of the gateway (you can find it with the command `route`). Next, she starts the DNS spoofing attack with the following command:

```
ettercap -T -M arp:remote /IPaddr_host_Bob// /IPaddr_GW// -P dns_spoof
```

5. Bob tries to connect to the web site www.polito.it.

What does Bob see in his web browser?

→

What could have happened if the attack didn't work? Make your assumptions and confront them with what we have indicated in the initial note.

→

What does Alice see in the `ettercap` window?

→

Check out the configuration of the DNS plug-in, in the file `/etc/ettercap/etter.dns`.

How can an attacker obtain similar results (as the ones described in this section), even though the attacker is not close to the victim?

→