PENETRATION TESTING

BANNER GRABBING

BANNER GRABBING

In the previous lesson, we examined the main network scanning techniques. Now it's time to identify what type of service is running on a specific port.

BANNER GRABBING

This information will be useful to us in the next phase where we will look for vulnerabilities. In particular, the outdated version of a service could be exploited by a potential hacker.

BANNER GRABBING

We will start from the services normally associated with standard ports, and then move on the ones linked to unconventional ports.

METASPLOITABLE 2

METASPLOITABLE2

Metasploitable 2 is an intentionally vulnerable environment for penetration testers training and security scanners evaluation

Developed by Rapid7, the same vendor of the Metasploit security framework

Predecessor: Metasploitable

Successor: Metasploitable 3

METASPLOITABLE 2

We import a docker image of Metasploitable 2 in GNS3

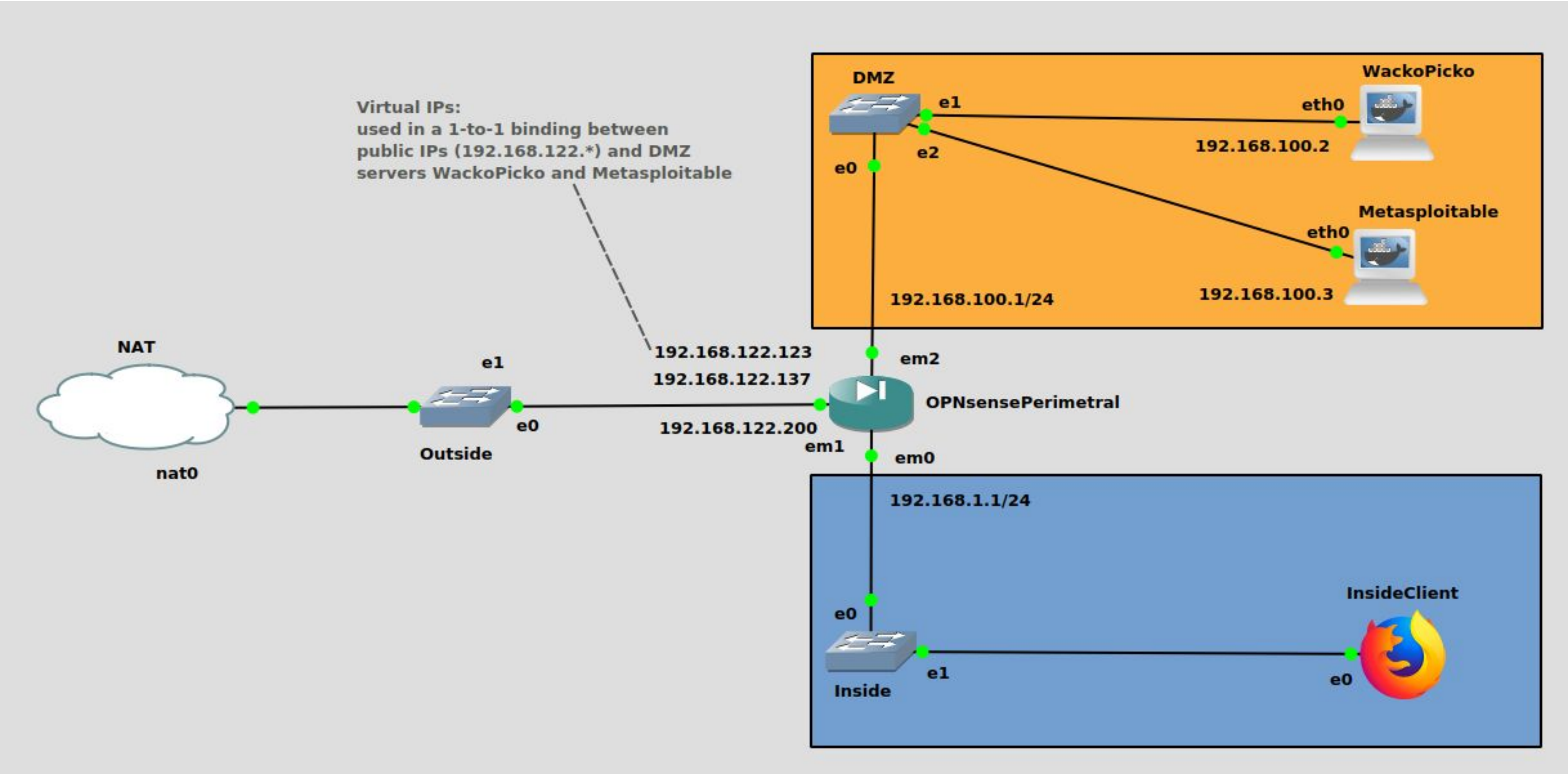
peakkk/metasploitable

Exercise (~15’)

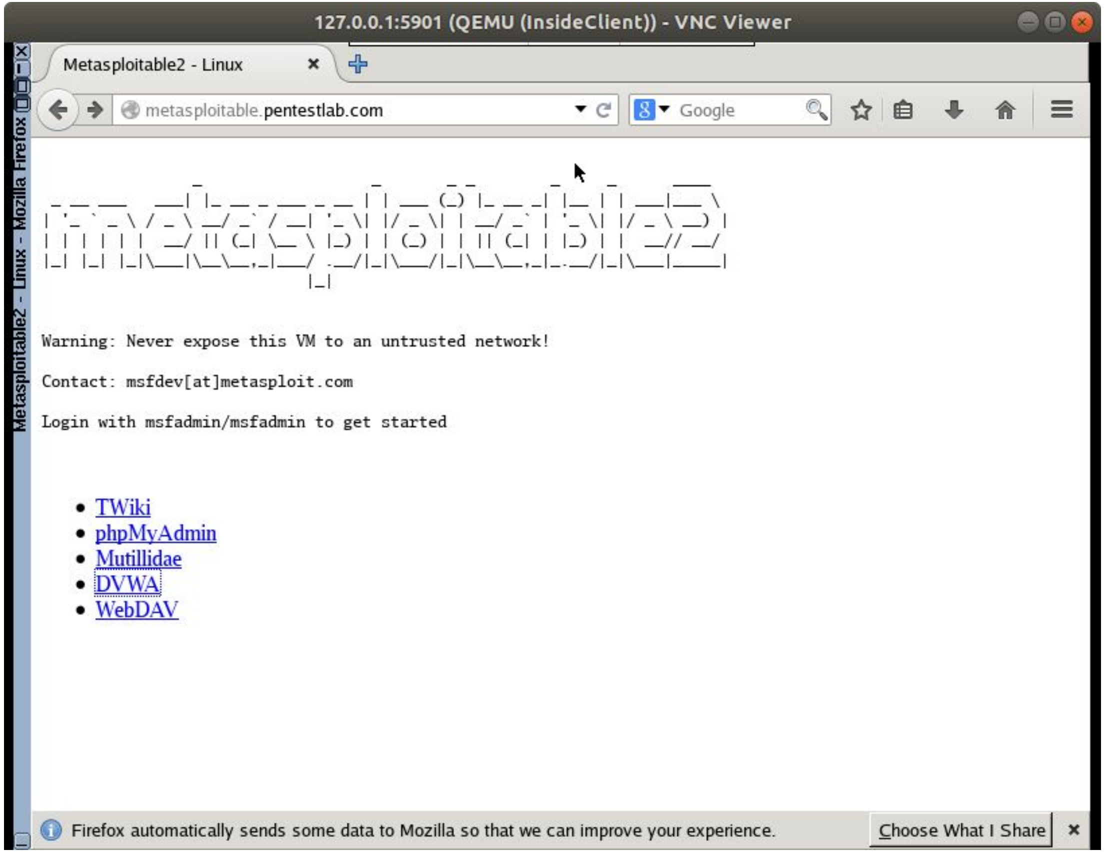
Carry out the following steps

1. instantiate Metasploitable 2 and connect it to the DMZ
2. assign a static IP address to Metasploitable 2
3. Create a public, virtual IP and bind it to Metasploitable 2
4. Define a symbolic name in the DNS (e.g., metasploitable.pentestlab.com)

METASPLOITABLE 2



METASPLOITABLE 2



127.0.0.1:5901 (QEMU (InsideClient)) - VNC Viewer
Metasploitable2 - Linux - Mozilla Firefox
Metasploitable2 - Linux
metasploitable.pentestlab.com
Google

```
                _                  _       _ _        _     _      ____  
 _ __ ___   ___| |_ __ _ ___ _ __ | | ___ (_) |_ __ _| |__ | | ___|___ \ 
| '_ ` _ \ / _ \ __/ _` / __| '_ \| |/ _ \| | __/ _` | '_ \| |/ _ \ __) |
| | | | | |  __/ || (_| \__ \ |_) | | (_) | | || (_| | |_) | |  __// __/ 
|_| |_| |_|\___|\__\__,_|___/ .__/|_|\___/|_|\__\__,_|_.__/|_|\___|_____|
                            |_|                                          
```

Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started

- TWiki
- phpMyAdmin
- Mutillidae
- DVWA
- WebDAV

Firefox automatically sends some data to Mozilla so that we can improve your experience.

BANNER VISUALIZATION
HTTP SERVER

BANNER GRABBING

Metasploitable hosts (at least) one web server that we already connected to

At this point, we must be able to grab the banner of the web server so that we can detect its type and version

We can open a connection with the web server using telnet by typing

```
telnet IP PORT
COMMAND
```

BANNER GRABBING

The telnet commands that we can submit refer to HTTP methods

For instance, to GET the rood document

```
GET / HTTP/1.1
HOST: 127.0.0.1
```

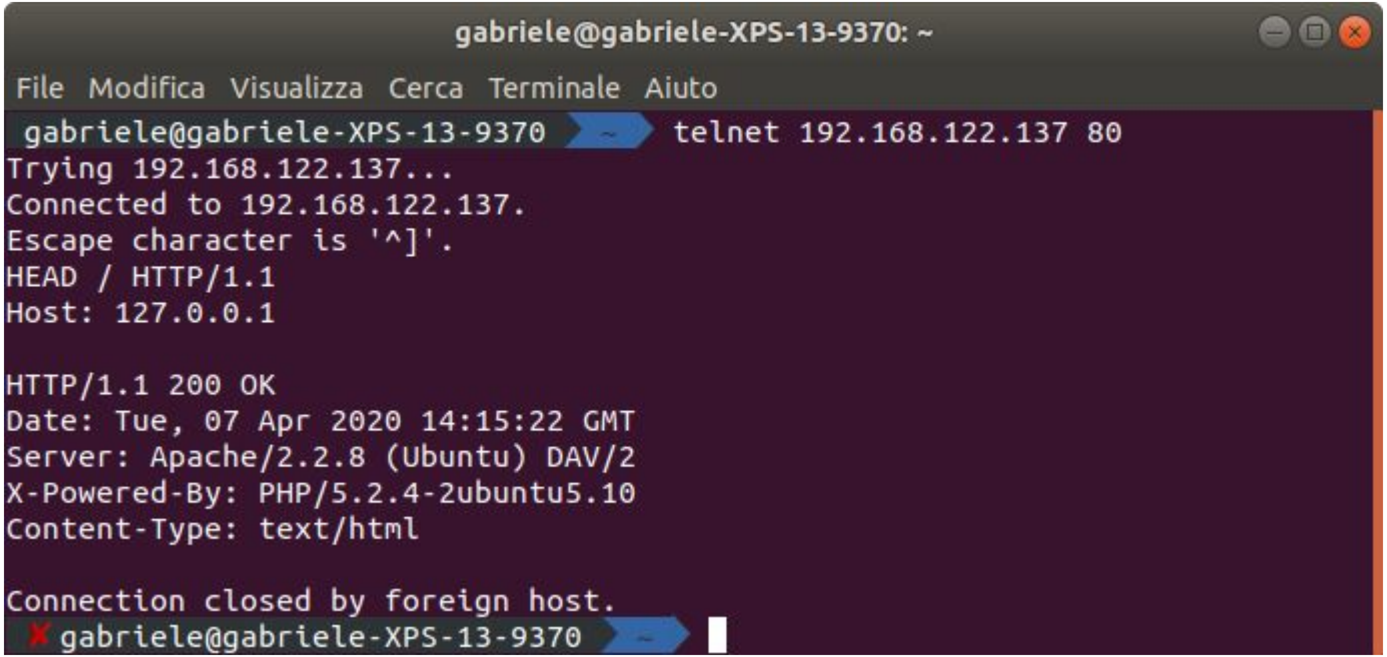
To only retrieve the HTTP header

```
HEAD / HTTP/1.1
HOST: 127.0.0.1
```

Exercise (~2’)

Find the server name and version

BANNER GRABBING

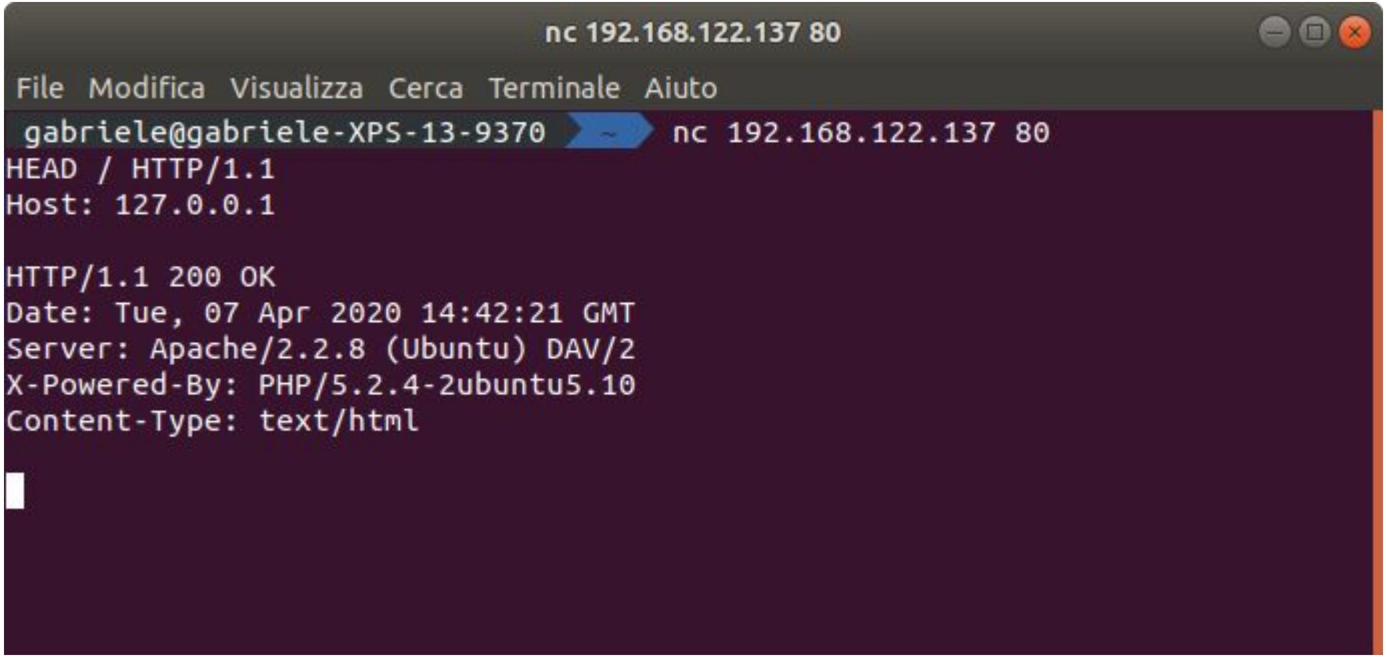


gabriele@gabriele-XPS-13-9370: ~
File Modifica Visualizza Cerca Terminale Aiuto

```
 gabriele@gabriele-XPS-13-9370  ~  telnet 192.168.122.137 80
Trying 192.168.122.137...
Connected to 192.168.122.137.
Escape character is '^]'.
HEAD / HTTP/1.1
Host: 127.0.0.1

HTTP/1.1 200 OK
Date: Tue, 07 Apr 2020 14:15:22 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Content-Type: text/html

Connection closed by foreign host.
 ✘ gabriele@gabriele-XPS-13-9370  ~ 
```

BANNER GRABBING

The same result can be obtained with netcat (we will see more about netcat)



```
gabriele@gabriele-XPS-13-9370  ~  nc 192.168.122.137 80
HEAD / HTTP/1.1
Host: 127.0.0.1

HTTP/1.1 200 OK
Date: Tue, 07 Apr 2020 14:42:21 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Content-Type: text/html
```

BANNER GRABBING

We have captured the banner of our web server.

We can now identify the type of service and its version.

This information will be useful during the vulnerability assessment phase.

FTP BANNER GRABBING WITH NMAP

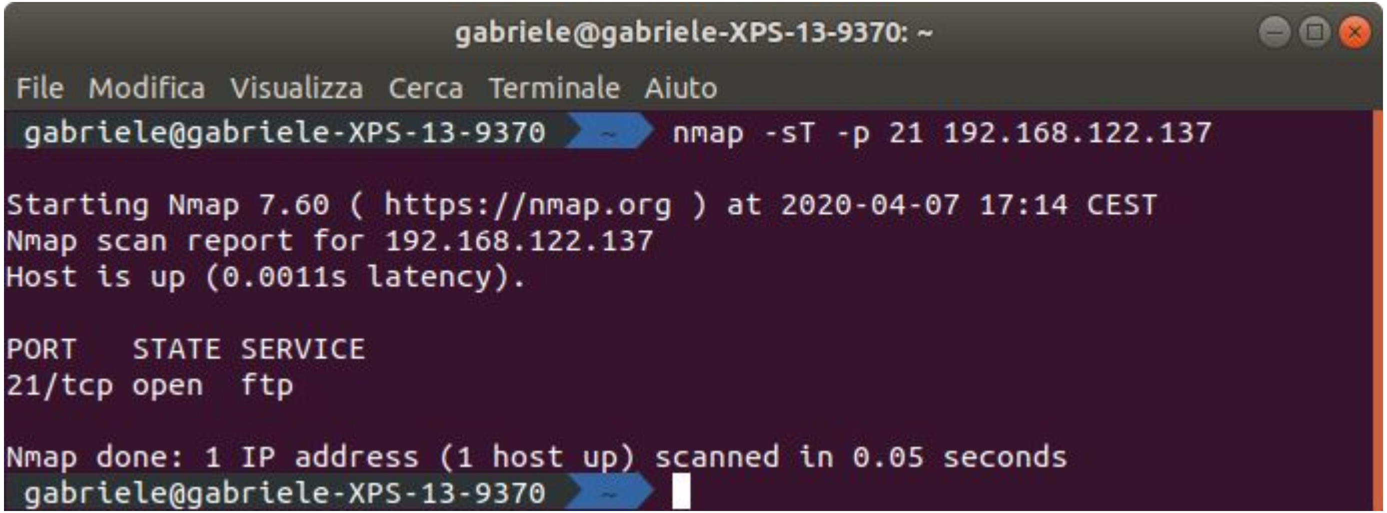
BANNER GRABBING

Metasploitable 2 runs a FTP service

Exercise (~2’)

Use what we have learned about network scanning and find it with nmap

BANNER GRABBING

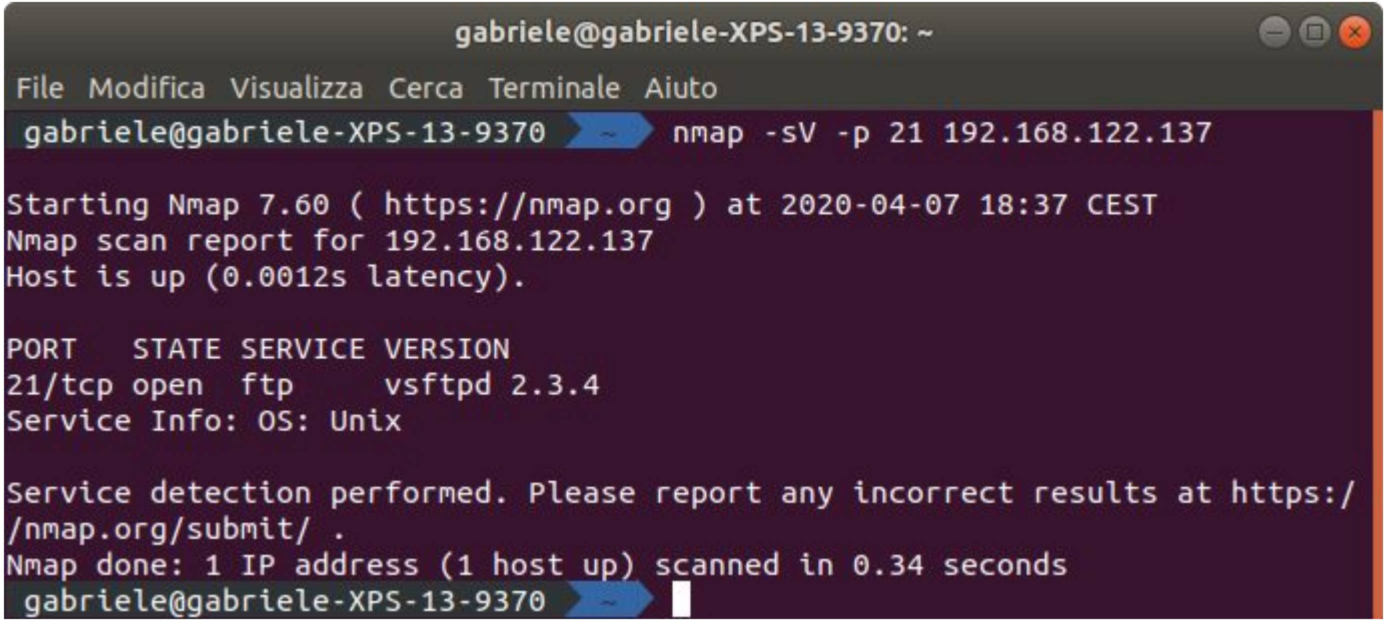


gabriele@gabriele-XPS-13-9370: ~
File Modifica Visualizza Cerca Terminale Aiuto

```
gabriele@gabriele-XPS-13-9370  ~  nmap -sT -p 21 192.168.122.137

Starting Nmap 7.60 ( https://nmap.org ) at 2020-04-07 17:14 CEST
Nmap scan report for 192.168.122.137
Host is up (0.0011s latency).

PORT   STATE SERVICE
21/tcp open  ftp

Nmap done: 1 IP address (1 host up) scanned in 0.05 seconds
gabriele@gabriele-XPS-13-9370  ~
```

BANNER GRABBING

To detect the FTP server version we can use option -sV



gabriele@gabriele-XPS-13-9370: ~
File Modifica Visualizza Cerca Terminale Aiuto

```
gabriele@gabriele-XPS-13-9370  ~  nmap -sV -p 21 192.168.122.137

Starting Nmap 7.60 ( https://nmap.org ) at 2020-04-07 18:37 CEST
Nmap scan report for 192.168.122.137
Host is up (0.0012s latency).

PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 2.3.4
Service Info: OS: Unix

Service detection performed. Please report any incorrect results at https:/
/nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.34 seconds
gabriele@gabriele-XPS-13-9370  ~  
```

BANNER GRABBING

Note that some system administrators may decide to obfuscate the banner for a certain service

The default vsFTP banner can be replaced by editing /etc/[vsftpd/]vsftpd.conf

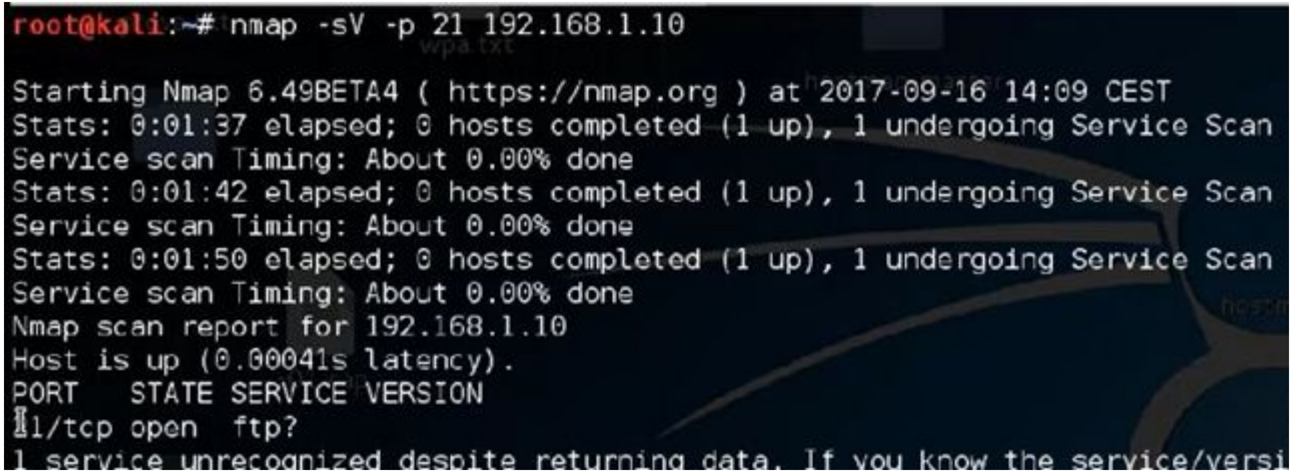
There you decomment and modify the line

```
    #ftpd_banner=Welcome to blah FTP service
```

BANNER GRABBING

Now we should no longer be able to detect the version of the service with Nmap

BANNER GRABBING



```
root@kali:~# nmap -sV -p 21 192.168.1.10

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2017-09-16 14:09 CEST
Stats: 0:01:37 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 0.00% done
Stats: 0:01:42 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 0.00% done
Stats: 0:01:50 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 0.00% done
Nmap scan report for 192.168.1.10
Host is up (0.00041s latency).
PORT   STATE SERVICE VERSION
21/tcp open  ftp?
1 service unrecognized despite returning data. If you know the service/versi
```

BANNER GRABBING

Nmap was able to understand that port 21 is open. However, it does not provide any information about the version of the service running.

FTP BANNER GRABBING WITH METASPLOIT

BANNER GRABBING

Now let's try grabbing a banner with Metasploit, a tool that we will explore in depth in the next chapters

Metasploit is a security framework supporting all the pentesting phases

Metasploit has a number of modules that allow you to perform several activities, such as banner grabbing

BANNER GRABBING

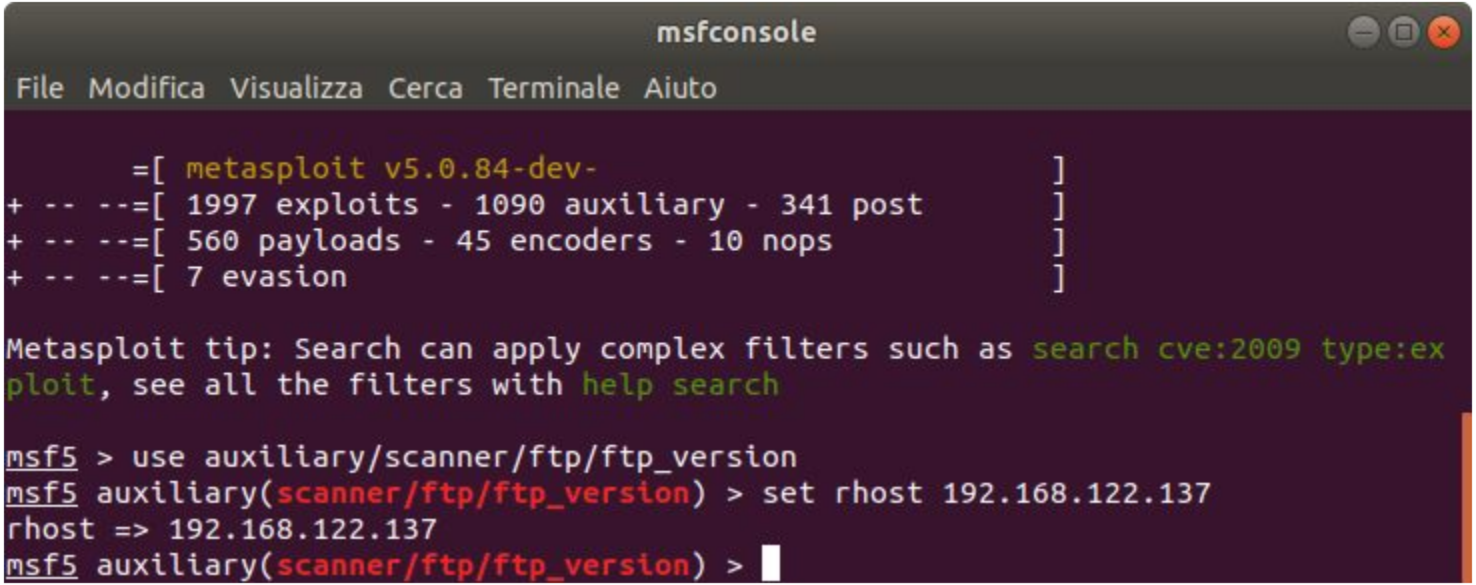
Install the Metasploit framework from https://www.metasploit.com/download

Pre-installed in Kali linux

We start Metasploit by launching the "msf" or "msfconsole" command from a terminal.

Then we type the following command:

BANNER GRABBING



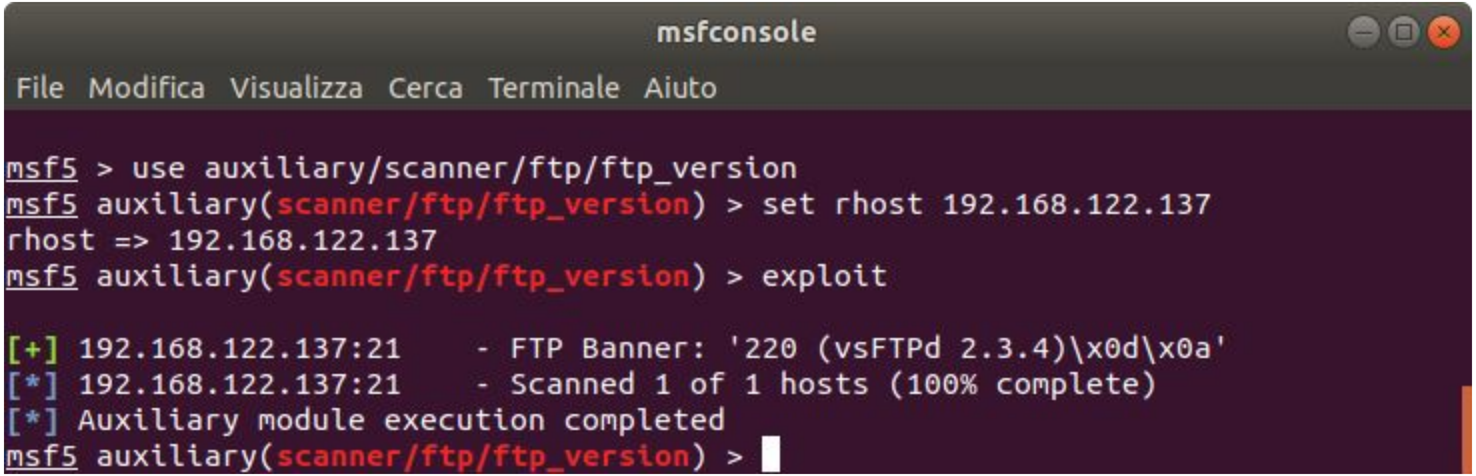
```
       =[ metasploit v5.0.84-dev-                         ]
+ -- --=[ 1997 exploits - 1090 auxiliary - 341 post       ]
+ -- --=[ 560 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 7 evasion                                       ]

Metasploit tip: Search can apply complex filters such as search cve:2009 type:ex
ploit, see all the filters with help search

msf5 > use auxiliary/scanner/ftp/ftp_version
msf5 auxiliary(scanner/ftp/ftp_version) > set rhost 192.168.122.137
rhost => 192.168.122.137
msf5 auxiliary(scanner/ftp/ftp_version) >
```

BANNER GRABBING

We set the IP address of the target machine running the FTP service as remote host (rhost).

Then, we can run the "exploit" command and then start the scan.

BANNER GRABBING



```
msf5 > use auxiliary/scanner/ftp/ftp_version
msf5 auxiliary(scanner/ftp/ftp_version) > set rhost 192.168.122.137
rhost => 192.168.122.137
msf5 auxiliary(scanner/ftp/ftp_version) > exploit

[+] 192.168.122.137:21    - FTP Banner: '220 (vsFTPd 2.3.4)\x0d\x0a'
[*] 192.168.122.137:21    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/ftp/ftp_version) > 
```

BANNER GRABBING

The scan is quickly completed, and the result obtained informs us of the presence of a vsFTP server. We grabbed the banner once again.

FTP BANNER GRABBING WITH NETCAT

BANNER GRABBING

NETCAT is another useful tool used for establishing TCP/UDP connections

It is often referred to as the network swiss army knife (see https://en.wikipedia.org/wiki/Netcat)

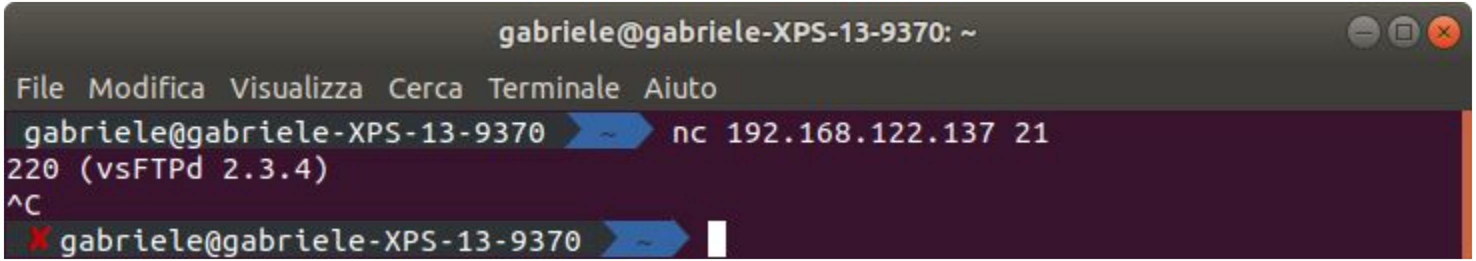
Clearly we can use it for banner grabbing (we already did for HTTP)

Below is the command used to grab the FTP banner:

BANNER GRABBING

Below is the command used to grab the FTP banner

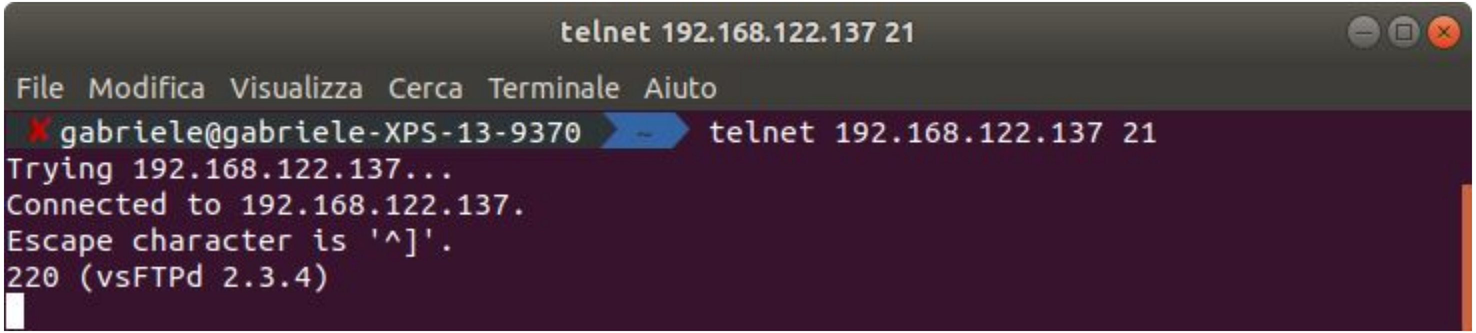
Briefly, we just need to connect


gabriele@gabriele-XPS-13-9370: ~
File Modifica Visualizza Cerca Terminale Aiuto

```
 gabriele@gabriele-XPS-13-9370  ~  nc 192.168.122.137 21
220 (vsFTPd 2.3.4)
^C
 ✘ gabriele@gabriele-XPS-13-9370  ~ 
```

BANNER GRABBING

The very same can be done with telnet


telnet 192.168.122.137 21
File Modifica Visualizza Cerca Terminale Aiuto

```
gabriele@gabriele-XPS-13-9370  ~  telnet 192.168.122.137 21
Trying 192.168.122.137...
Connected to 192.168.122.137.
Escape character is '^]'.
220 (vsFTPd 2.3.4)
```

NMAP SERVICE PROBES

How does NMap detects services?

It uses a long list of **probes**, i.e., rules stating which message should be sent to test a service and how to parse the output

Parsing is based on **regular expressions** and **capture groups** as in this example

```
match ftp m|^220 \(vsFTPd ([-.\w]+)\)\r\n$|
p/vsftpd/ v/$1/ o/Unix/ cpe:/a:vsftpd:vsftpd:$1/
```

OPERATING SYSTEM DETECTION

RELEVANT CONCEPT

In addition to detecting a certain running service, it is also important to know the operating system present on the target machine.

BANNER GRABBING

We can follow two different procedures:

- Active mode.
- Passive mode.

In the active mode, we interact directly with the target.

Nmap is a tool commonly used in active mode.

BANNER GRABBING

On the other hand, the passive mode silently observes the network traffic.

Based on the characteristics of each operating system, we can obtain fairly precise information.

A tool that works in this mode is "p0f" (https://it.wikipedia.org/wiki/P0f)

Notice that in most cases passive mode relies on a sniffer (not always possible)

OS DETECTION WITH NMAP

BANNER GRABBING

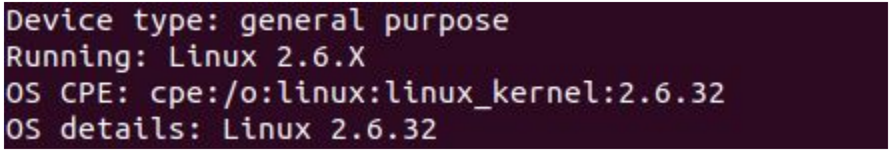
Let's see how to actively detect the operating system of a certain machine using Nmap. The option to use is "-O", so this command will be the command we need to execute:

```
nmap -O -v 192.168.122.137
```

BANNER GRABBING

By running this command, we will examine the open ports and try to detect the operating system.

The result is the following:



```
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6.32
OS details: Linux 2.6.32
```

BANNER GRABBING

Nmap was able to identify that the operating system in use is likely to be a Linux distribution and specifically version kernel version 2.6.32

Not perfectly accurate (the actual kernel was 4.15.0)

BANNER GRABBING

XPROBE is another tool useful for detecting the operating system. This is the command we should execute:

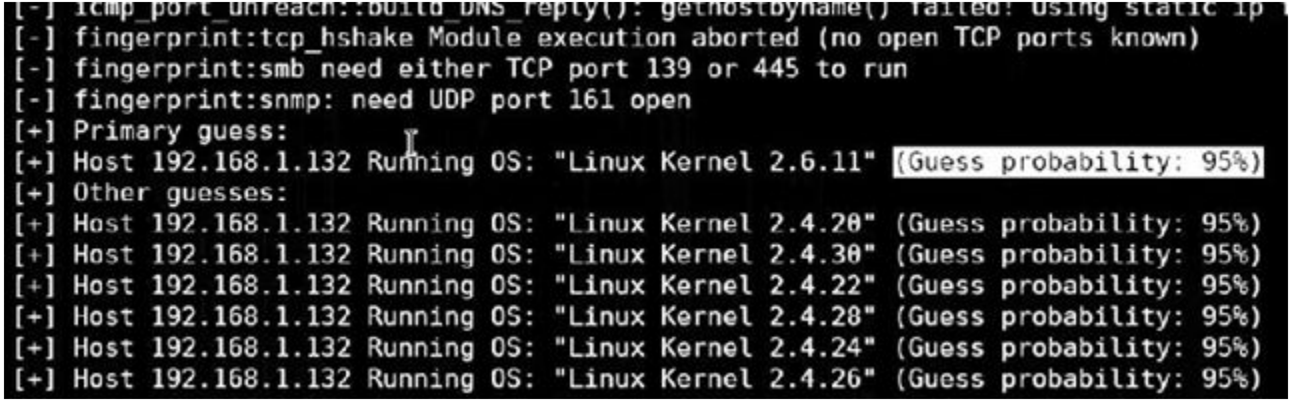
```
xprobe2 ADDRESS
```

XPROBE returns a list of candidate OSes (with a probability value) and is rather accurate

NOTICE: the current version has a known bug and prints garbage chars!

BANNER GRABBING

You should see the following results:



```
[-] icmp_port_unreach::build_DNS_reply(): gethostbyname() failed! Using static ip
[-] fingerprint:tcp_hshake Module execution aborted (no open TCP ports known)
[-] fingerprint:smb need either TCP port 139 or 445 to run
[-] fingerprint:snmp: need UDP port 161 open
[+] Primary guess:
[+] Host 192.168.1.132 Running OS: "Linux Kernel 2.6.11" (Guess probability: 95%)
[+] Other guesses:
[+] Host 192.168.1.132 Running OS: "Linux Kernel 2.4.20" (Guess probability: 95%)
[+] Host 192.168.1.132 Running OS: "Linux Kernel 2.4.30" (Guess probability: 95%)
[+] Host 192.168.1.132 Running OS: "Linux Kernel 2.4.22" (Guess probability: 95%)
[+] Host 192.168.1.132 Running OS: "Linux Kernel 2.4.28" (Guess probability: 95%)
[+] Host 192.168.1.132 Running OS: "Linux Kernel 2.4.24" (Guess probability: 95%)
[+] Host 192.168.1.132 Running OS: "Linux Kernel 2.4.26" (Guess probability: 95%)
```

OS DETECTION WITH P0F

BANNER GRABBING

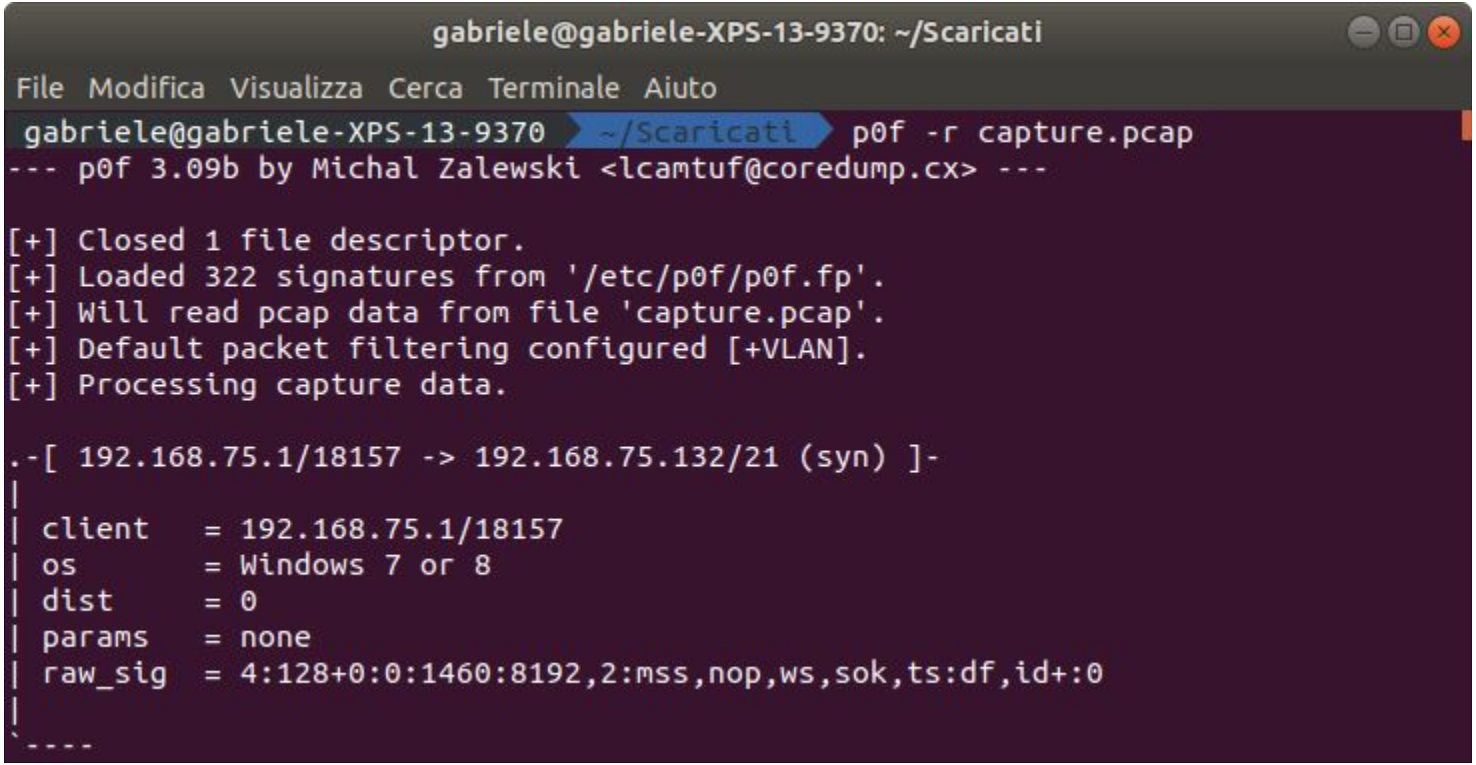
As anticipated, this tool allows to perform a passive operating system detection. In this case, we do not need to interact directly with the target machine.

We need to capture some network traffic, so that P0f can complete the detection process.

P0f can work with both live captures and recorded sessions.

We can analyze a target pcap and find the OS of machine 192.168.75.1

BANNER GRABBING

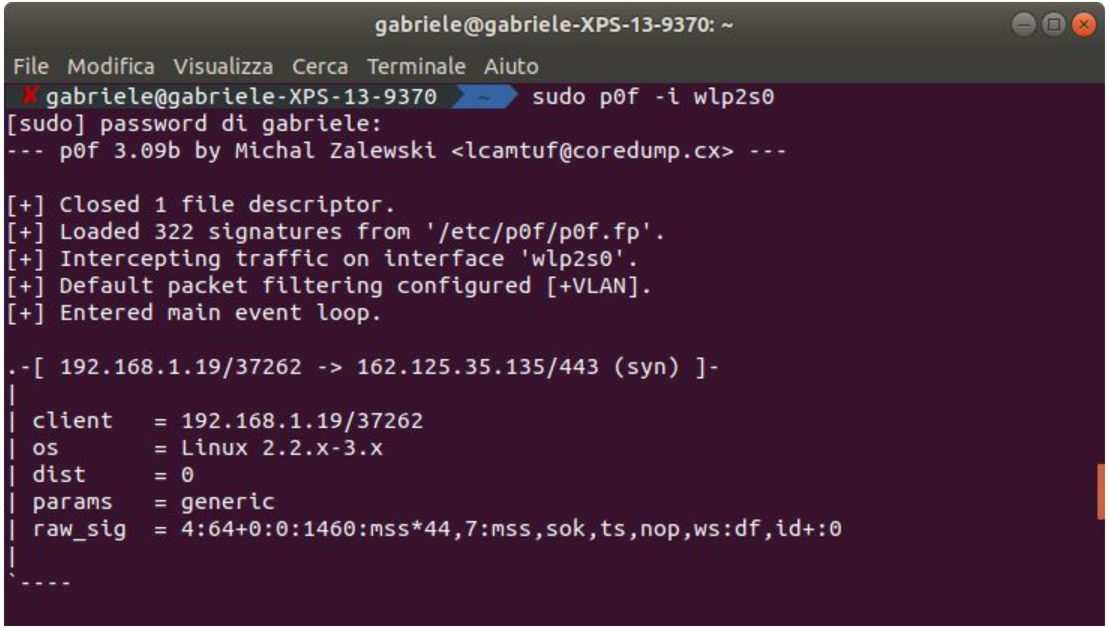


gabriele@gabriele-XPS-13-9370: ~/Scaricati
File Modifica Visualizza Cerca Terminale Aiuto

```
 gabriele@gabriele-XPS-13-9370  ~/Scaricati  p0f -r capture.pcap
--- p0f 3.09b by Michal Zalewski <lcamtuf@coredump.cx> ---

[+] Closed 1 file descriptor.
[+] Loaded 322 signatures from '/etc/p0f/p0f.fp'.
[+] Will read pcap data from file 'capture.pcap'.
[+] Default packet filtering configured [+VLAN].
[+] Processing capture data.

.-[ 192.168.75.1/18157 -> 192.168.75.132/21 (syn) ]-
|
| client   = 192.168.75.1/18157
| os       = Windows 7 or 8
| dist     = 0
| params   = none
| raw_sig  = 4:128+0:0:1460:8192,2:mss,nop,ws,sok,ts:df,id+:0
|
`----
```

BANNER GRABBING

Live capture on a machine interface



gabriele@gabriele-XPS-13-9370: ~
File Modifica Visualizza Cerca Terminale Aiuto

```
 ✗ gabriele@gabriele-XPS-13-9370  ~  sudo p0f -i wlp2s0
[sudo] password di gabriele:
--- p0f 3.09b by Michal Zalewski <lcamtuf@coredump.cx> ---

[+] Closed 1 file descriptor.
[+] Loaded 322 signatures from '/etc/p0f/p0f.fp'.
[+] Intercepting traffic on interface 'wlp2s0'.
[+] Default packet filtering configured [+VLAN].
[+] Entered main event loop.

.-[ 192.168.1.19/37262 -> 162.125.35.135/443 (syn) ]-
|
| client   = 192.168.1.19/37262
| os       = Linux 2.2.x-3.x
| dist     = 0
| params   = generic
| raw_sig  = 4:64+0:0:1460:mss*44,7:mss,sok,ts,nop,ws:df,id+:0
|
`----
```

How does this work

Simply, network traffic generated by different OSes has some peculiarities

This is often due to default values

For instance, under most Linux systems the default TTL is 64, while Windows often uses 128