

FINGERPRINTING

APLICACIONES Y SISTEMA OPERATIVO



1. UN POCO DE TEORÍA

Tras haber visto las técnicas de host discover y port scanning en una red a la que estamos conectados, ya estamos en condiciones de poder analizar estos equipos a un nivel más profundo. A la técnica que permite averiguar características de equipos en una red por medio de análisis de paquetería se le denomina **fingerprinting**.

Fingerprinting nace de la palabra inglesa fingerprint (huella dactilar) ya que es una técnica de poder conocer las características de un equipo solo por las “huellas” que va dejando en sus paquetes.

El fingerprinting se puede clasificar en 2 grandes tipos según su enfoque:

- **Activo:** interroga a la máquina con paquetes específicos para analizar las respuestas y establecer un posible sistema operativo.
- **Pasivo:** escucha información que se intercambian y la analiza.

El fingerprinting, a su vez, puede ser aplicado a 2 tipos de información: conocer el sistema operativo que corre en la máquina o conocer las aplicaciones (nombre, versión...) que corren en cierto servicio. Es de suma importancia para conocer posibles **vulnerabilidades**, ya que si sabes que una máquina corre un determinado servidor con determinada versión en cierto sistema operativo, puedes aplicarle un exploit y acabar con el.

Para este artículo trabajaremos sobre varios equipos Linux con un servidor Apache en puerto 80, servidor Apache Tomcat en el puerto 8080 y un servidor OpenSSH sobre el puerto 22, además de hacer pruebas con un servidor de la web de la Universidad de la Coruña (www.udc.es).

Para el fingerprinting se usarán las siguientes herramientas, que serán explicadas en la parte práctica:

- **OS Fingerprinting:**
 - p0f
 - xprobe2
 - nmap
- **Fingerprinting de aplicaciones**
 - NetCat (nc)
 - nmap
 - httpprint
 - whatweb
 - shodan (extensión para firefox o chrome)

A mayores usaré dig, una herramienta que proporciona la IP que devuelve la DNS dado un dominio. También usaremos un navegador web.

Más acerca de dig: <https://linux.die.net/man/1/dig>

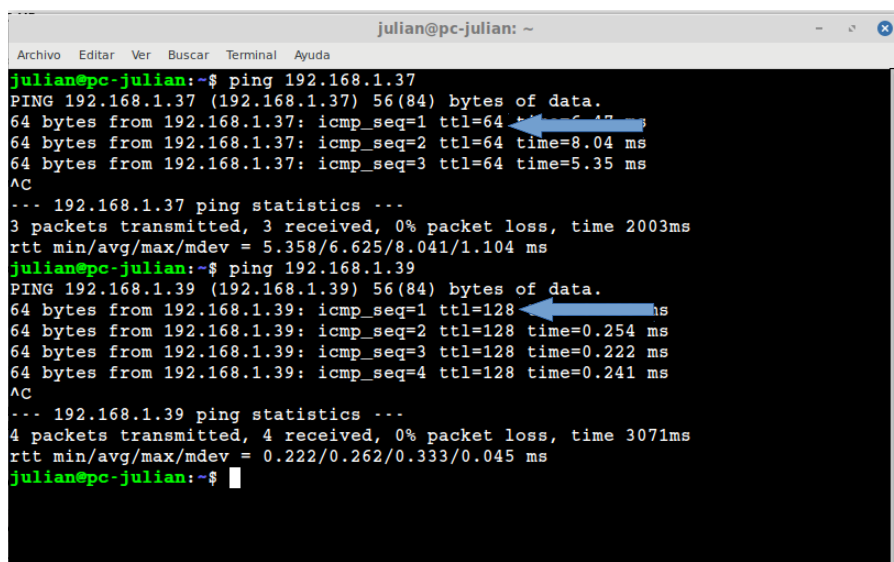
2. PASANDO DIRECTAMENTE A LA PRÁCTICA

➤ OS FINGERPRINT

El método para averiguar el sistema operativo se basa en analizar paquetes, sus cabeceras y su comportamiento, para lograr averiguarlo. Algunas de estas características son:

- TTL, por ejemplo, en un ping: en Windows vale 128, en Debian 48, en el resto de Linux y BSD 64 y en cisco 255.
- MSS
- Flag NOP de TCP
- Window Size

Ping a Ubuntu (192.168.1.37) y a Windows (192.168.1.39):



```
julian@pc-julian: ~  
Archivo  Editar  Ver  Buscar  Terminal  Ayuda  
julian@pc-julian:~$ ping 192.168.1.37  
PING 192.168.1.37 (192.168.1.37) 56(84) bytes of data.  
64 bytes from 192.168.1.37: icmp_seq=1 ttl=64 time=6.47 ms  
64 bytes from 192.168.1.37: icmp_seq=2 ttl=64 time=8.04 ms  
64 bytes from 192.168.1.37: icmp_seq=3 ttl=64 time=5.35 ms  
AC  
--- 192.168.1.37 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2003ms  
rtt min/avg/max/mdev = 5.358/6.625/8.041/1.104 ms  
julian@pc-julian:~$ ping 192.168.1.39  
PING 192.168.1.39 (192.168.1.39) 56(84) bytes of data.  
64 bytes from 192.168.1.39: icmp_seq=1 ttl=128 time=0.254 ms  
64 bytes from 192.168.1.39: icmp_seq=2 ttl=128 time=0.222 ms  
64 bytes from 192.168.1.39: icmp_seq=3 ttl=128 time=0.222 ms  
64 bytes from 192.168.1.39: icmp_seq=4 ttl=128 time=0.241 ms  
AC  
--- 192.168.1.39 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3071ms  
rtt min/avg/max/mdev = 0.222/0.262/0.333/0.045 ms  
julian@pc-julian:~$
```

Podríamos ser nosotros mismos los que analizáramos dicho comportamiento, pero existen varias herramientas que lo hacen por nosotros. En este caso, yo elegí las 3 que mejor me funcionaron siempre y que son útiles en cualquier caso.

- **NMAP**

Seguro la recordareis de todo este apartado de análisis de entorno, y no podía faltarle una opción para fingerprint de sistema operativo. La opción de **Nmap -O** permite a esta herramienta el envío de tráfico adaptado a puertos para analizar el comportamiento y dar un resultado estimado de que sistema está corriendo.

```

julian@pc-julian: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
julian@pc-julian:~$ sudo nmap -O 192.168.1.39
[sudo] password for julian:

Starting Nmap 7.60 ( https://nmap.org ) at 2018-03-24 12:42 CET
Nmap scan report for 192.168.1.39
Host is up (0.00051s latency).
Not shown: 996 filtered ports
PORT      STATE SERVICE
135/tcp    open  msrpc
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
4000/tcp   open  remoteanything
MAC Address: 08:00:27:4B:12:3A (Oracle VirtualBox virtual NIC)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose|specialized|phone
Running: Microsoft Windows 2008|8.1|7|Phone|Vista
OS CPE: cpe:/o:microsoft:windows_server_2008:r2 cpe:/o:microsoft:windows_8.1 cpe:/o:microsoft:windows_7::-:professional cpe:/o:microsoft:windows_8 cpe:/o:microsoft:windows_7 cpe:/o:microsoft:windows cpe:/o:microsoft:windows_vista::- cpe:/o:microsoft:windows_vista::spl
OS details: Microsoft Windows Server 2008 R2 or Windows 8.1, Microsoft Windows 7 Professional or Windows 8, Microsoft Windows Embedded Standard 7, Microsoft Windows Phone 7.5 or 8.0, Microsoft Windows Vista SP0 or SP1, Windows Server 2008 SP1, or Windows 7, Microsoft Windows Vista SP2, Windows 7 SP1, or Windows Server 2008
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 40.18 seconds
julian@pc-julian:~$
```

Podemos ver que, ante la máquina que hicimos ping anteriormente, nmap estima que corre un Windows superior a XP, y no se equivoca. Como es lógico, muchas veces las herramientas son incapaces de poder identificar exactamente el sistema operativo, y dan un rango de ellos como en este caso.

A continuación, te dejo otro ejemplo de un Linux Ubuntu con kernel 4.4:

```

julian@pc-julian: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
julian@pc-julian:~$ sudo nmap -O 192.168.1.37
[sudo] password for julian:

Starting Nmap 7.60 ( https://nmap.org ) at 2018-03-24 13:00 CET
Nmap scan report for 192.168.1.37
Host is up (0.0052s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 00:0B:81:A0:E7:10 (Kaparel)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.8
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 67.49 seconds
julian@pc-julian:~$
```

Como vemos, cuando trabajamos sobre una versión de Linux, nmap prefiere hablar de la versión de kernel y no de la distribución en sí.

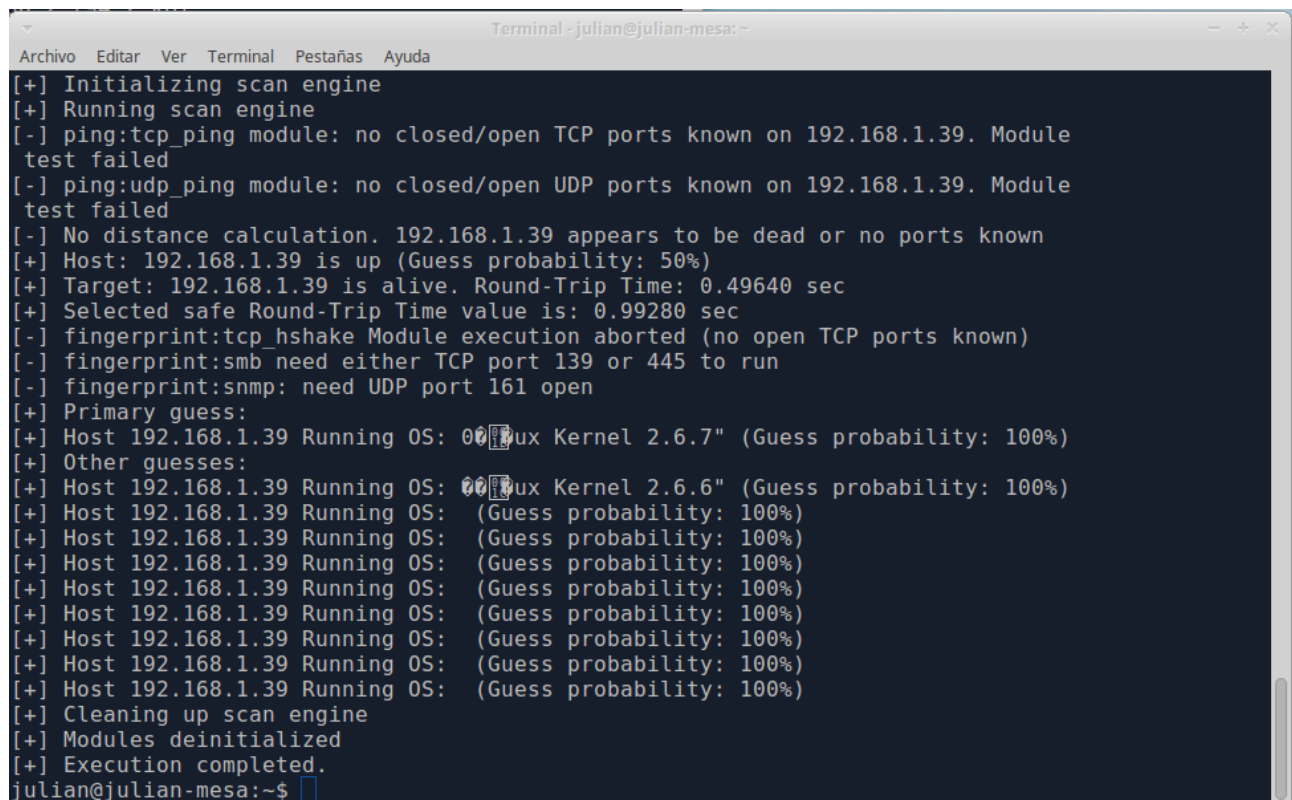
- **XPROBE2**

Xprobe2, al igual que Nmap, hace un fingerprinting **activo**. Se encarga de enviar paquetería al host para averiguar su sistema operativo.

Xprobe2 viene por defecto en Kali y en la mayoría de repositorios de gestores de paquetes.

La forma de ejecutarlo (con root) es sencillo:

xprobe2 [ip_host]



```
[+] Initializing scan engine
[+] Running scan engine
[-] ping:tcp_ping module: no closed/open TCP ports known on 192.168.1.39. Module
test failed
[-] ping:udp_ping module: no closed/open UDP ports known on 192.168.1.39. Module
test failed
[-] No distance calculation. 192.168.1.39 appears to be dead or no ports known
[+] Host: 192.168.1.39 is up (Guess probability: 50%)
[+] Target: 192.168.1.39 is alive. Round-Trip Time: 0.49640 sec
[+] Selected safe Round-Trip Time value is: 0.99280 sec
[-] 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 192.168.1.39 Running OS: 00Linux Kernel 2.6.7" (Guess probability: 100%)
[+] Other guesses:
[+] Host 192.168.1.39 Running OS: 00Linux Kernel 2.6.6" (Guess probability: 100%)
[+] Host 192.168.1.39 Running OS: (Guess probability: 100%)
[+] Host 192.168.1.39 Running OS: (Guess probability: 100%)
[+] Host 192.168.1.39 Running OS: (Guess probability: 100%)
[+] Host 192.168.1.39 Running OS: (Guess probability: 100%)
[+] Host 192.168.1.39 Running OS: (Guess probability: 100%)
[+] Host 192.168.1.39 Running OS: (Guess probability: 100%)
[+] Host 192.168.1.39 Running OS: (Guess probability: 100%)
[+] Host 192.168.1.39 Running OS: (Guess probability: 100%)
[+] Host 192.168.1.39 Running OS: (Guess probability: 100%)
[+] Cleaning up scan engine
[+] Modules deinitialized
[+] Execution completed.
julian@julian-mesa:~$
```

En este caso no pude usar máquinas virtuales, ya que con estas no funciona correctamente, así que lo hice desde mi sobremesa a mi portátil.

Una ventaja de xprobe2 es que muestra probabilidades ante los posibles sistemas operativos y que muestra su protocolo de actuación.

- **p0f**

De esta lista, el único que realiza un fingerprinting **pasivo**. El problema que ocasiona que haga este tipo de fingerprinting es que necesitamos establecer un **flujo de comunicación entre dispositivos**. A cambio, nos proporciona la ventaja de ser silencioso, solo escucha la conversación y analiza paquetes.

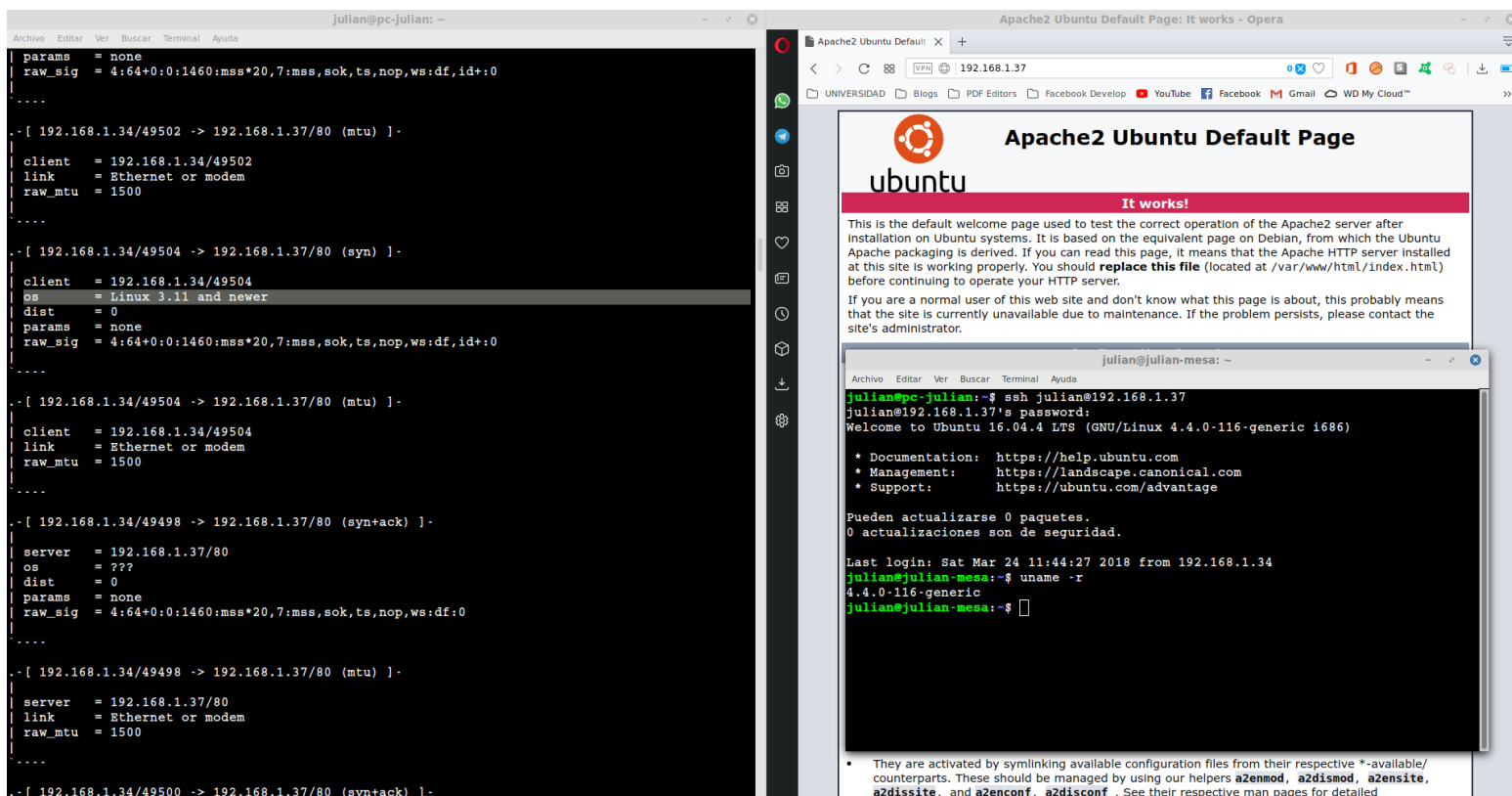
Viene instalado en Kali por defecto y también se encuentra en la mayoría de repositorios de gestores de paquetes, como apt.

Comando: **p0f -i [interfaz_red] -v**

<https://mihackeo.blogspot.com.es/>

Para mantener un flujo de paquetes, lo más sencillo es que la máquina esté corriendo un servicio al que nosotros podemos acceder (servidor web). De esta forma nosotros navegaremos por su web mientras p0f se dedica a analizar cabeceras y mostrarlas.

En este caso, he hecho un fingerprint a una máquina 192.168.1.37 que tiene un Linux en el que corre un Apache sobre el puerto 80, típico server web.



Podemos ver como muestra la paquetería e, incluso, el servidor web que corre:

```
.- [ 192.168.1.34/49498 -> 192.168.1.37/80 (http response) ] -  
|  
| server = 192.168.1.37/80  
| app = Apache 2.x  
| lang = none  
| params = none  
| raw_sig = 1:Date,Server,?Last-Modified,?ETag,Accept-Ranges=[bytes],?Vary,Content-Encoding=[  
gzip],?Content-Length,Keep-Alive=[timeout=5, max=100],Connection=[Keep-Alive],Content-Type::Ap  
ache/2.4.18 (Ubuntu)  
|  
| ----
```

Si navegamos durante cierto tiempo en la web acabaremos sacando el sistema operativo que corre en el.

CONCLUSIÓN:

Personalmente, yo uso mucho más **nmap** ya que es usado para todo el análisis de entorno y porque suele dar resultados más precisos y claros. Sin embargo, xprobe2 no suele defraudar en muchos casos y p0f es muy útiles en entornos donde no quieres ser monitorizado.

<https://mihackeo.blogspot.com.es/>

➤ FINGERPRINTING DE APLICACIONES

La técnica más usada aquí es el “**Banner Grabbing**” o **análisis de cabeceras**. Al establecer una conexión, se muestra una cabecera con los datos del servidor (bien de forma directa o indirecta). Se analiza la cabecera y sabemos que aplicación corre en el propio servidor.

He dejado fuera de este apartado dos conocidas herramientas como son **wfnet** y **fiddler**. El motivo es su complejo uso y su uso ÚNICAMENTE en Windows, ya que nosotros trabajamos principalmente el Linux. Creo que las que os voy mostrar son mucho más sencillas e igual o más útiles, además de multiplataforma.

- **NetCat**

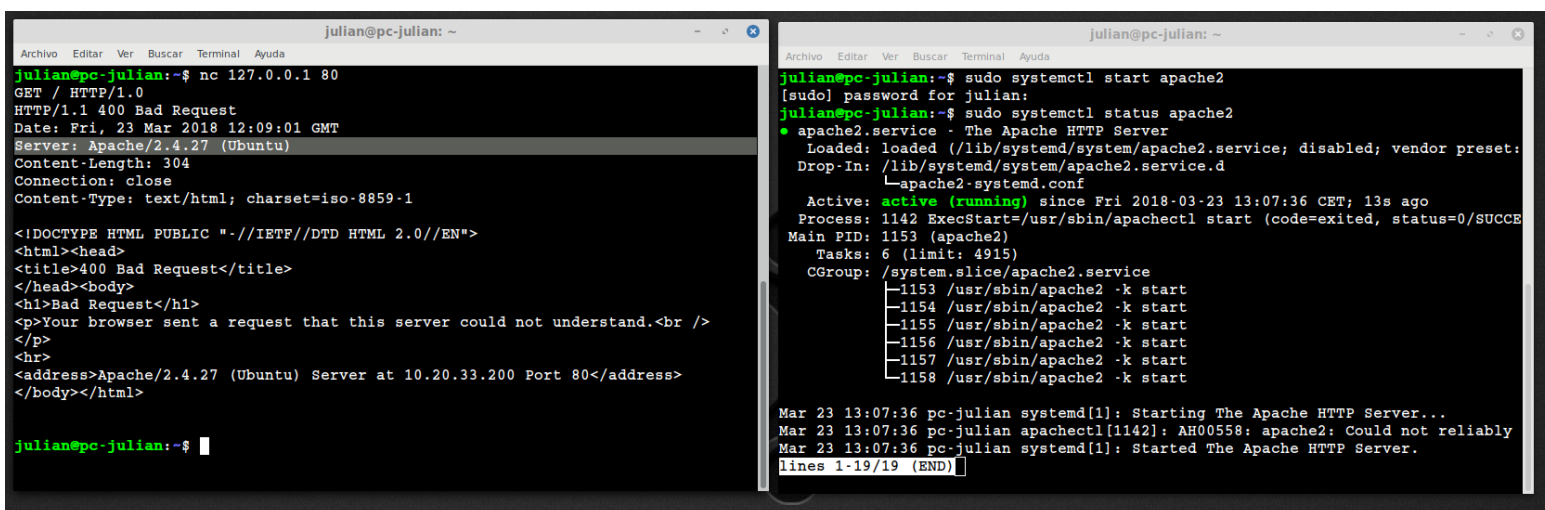
Esta herramienta no es especialmente para técnicas hacking, pero es muy útil para analizar servidores. Netcat se usa para establecer conexión con cualquier servidor en cualquier puerto y hacerle peticiones tras establecerla.

Netcat (nc) viene instalado por defecto en la mayoría de distribuciones linux.

Una vez establecida, el servidor muestra detalles de la conexión, y en muchos casos muestra nombre, que aplicación corre y que versión.

CASO 1 → APACHE PUERTO 80

En este caso habilito un servicio apache y me conecto a el:



The image contains two terminal screenshots. The left terminal shows a Netcat listener on port 80 receiving a GET request from 127.0.0.1, returning a 400 Bad Request status and an HTML body. The right terminal shows the user starting the Apache service with 'sudo systemctl start apache2', checking its status with 'sudo systemctl status apache2', and displaying the output which shows the service is active and running.

```
julian@pc-julian: ~  
julian@pc-julian:~$ nc 127.0.0.1 80  
GET / HTTP/1.0  
HTTP/1.1 400 Bad Request  
Date: Fri, 23 Mar 2018 12:09:01 GMT  
Server: Apache/2.4.27 (Ubuntu)  
Content-Length: 304  
Connection: close  
Content-Type: text/html; charset=iso-8859-1  
  
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">  
<html><head>  
<title>400 Bad Request</title>  
</head><body>  
<h1>Bad Request</h1>  
<p>Your browser sent a request that this server could not understand.<br />  
</p>  
<hr>  
<address>Apache/2.4.27 (Ubuntu) Server at 10.20.33.200 Port 80</address>  
</body></html>  
  
julian@pc-julian:~$  
  
julian@pc-julian:~$ sudo systemctl start apache2  
[sudo] password for julian:  
julian@pc-julian:~$ sudo systemctl status apache2  
● apache2.service - The Apache HTTP Server  
   Loaded: loaded (/lib/systemd/system/apache2.service; disabled; vendor preset: enabled)  
   Drop-In: /lib/systemd/system/apache2.service.d  
            └─apache2-systemd.conf  
   Active: active (running) since Fri 2018-03-23 13:07:36 CET; 13s ago  
     Process: 1142 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)  
    Main PID: 1153 (apache2)  
      Tasks: 6 (limit: 4915)  
   CGroup: /system.slice/apache2.service  
           └─1153 /usr/sbin/apache2 -k start  
             └─1154 /usr/sbin/apache2 -k start  
               └─1155 /usr/sbin/apache2 -k start  
                 └─1156 /usr/sbin/apache2 -k start  
                   └─1157 /usr/sbin/apache2 -k start  
                     └─1158 /usr/sbin/apache2 -k start  
  
Mar 23 13:07:36 pc-julian systemd[1]: Starting The Apache HTTP Server...  
Mar 23 13:07:36 pc-julian apachectl[1142]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.0.1  
Mar 23 13:07:36 pc-julian systemd[1]: Started The Apache HTTP Server.  
lines 1-19/19 (END)
```

Podemos ver que muestra el servidor y la versión, ya tendríamos la información para buscar un exploit.

Como vemos, Netcat funciona con el comando:

nc [ip_host] [puerto_servicio]

Posteriormente introducimos la petición al servidor.

Más acerca de Netcat: <https://linux.die.net/man/1/nc>

<https://mihackeo.blogspot.com.es/>

CASO 2 → SERVIDOR WEB DE LA WWW.UDC.ES

```

julian@pc-julian: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

julian@pc-julian:~$
julian@pc-julian:~$ nc www.udc.es 80
HEAD / HTTP/1.0

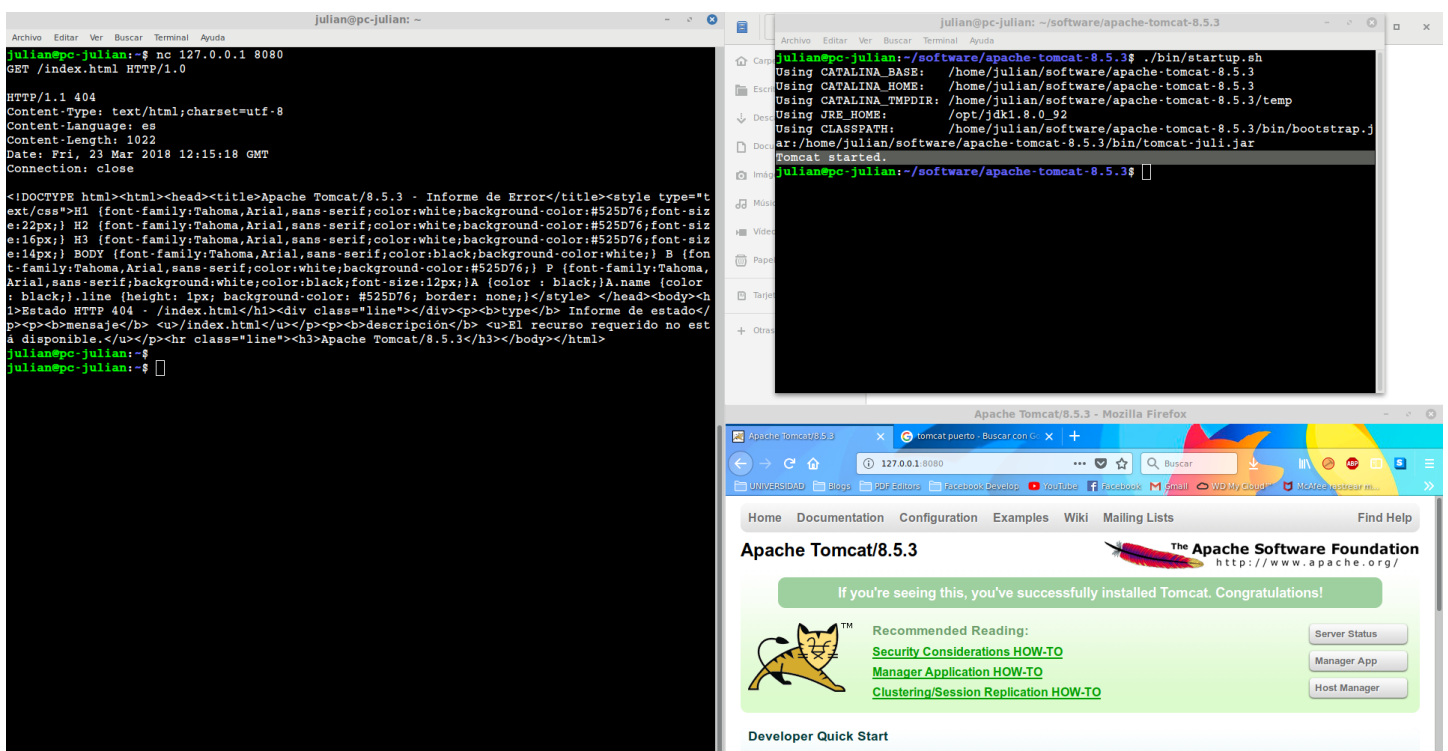
HTTP/1.1 400 Bad Request
Date: Fri, 23 Mar 2018 12:00:35 GMT
Server: Apache
Vary: Accept-Encoding
Content-Length: 226
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
</body></html>

julian@pc-julian:~$
```

En principio ya sabemos que corre un Apache. Profundizaremos con otras herramientas para conocer versión.

CASO 3 → SERVIDOR APACHE TOMCAT PUERTO 8080



The first screenshot shows a terminal window where a netcat listener on port 8080 receives a GET request to /index.html. The response is an HTTP 404 error with a detailed HTML error page from Apache Tomcat/8.5.3.

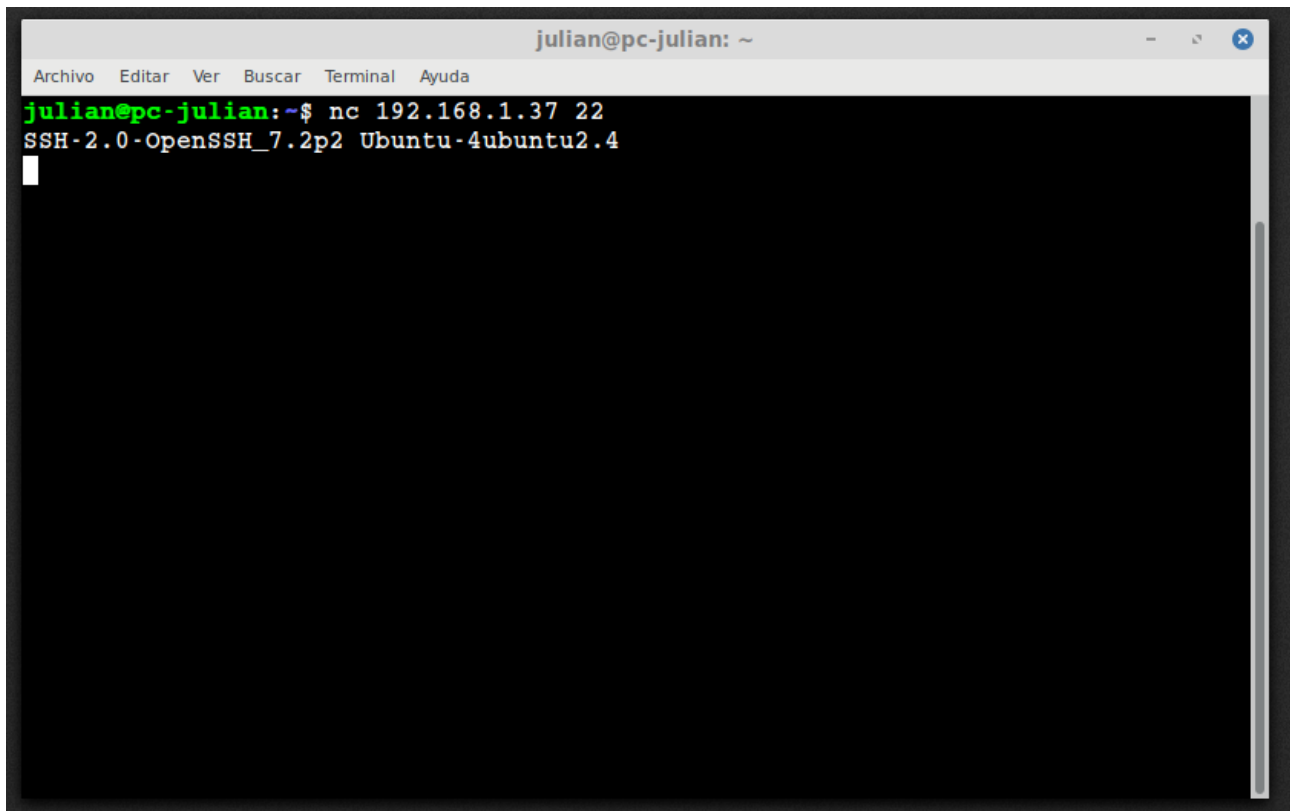
The second screenshot shows the terminal window where the Apache Tomcat 8.5.3 startup script is executed, showing the configuration and the message 'Tomcat started.'

The third screenshot shows a web browser displaying the Apache Tomcat 8.5.3 management page, which includes links to documentation, configuration, examples, and a 'Find Help' button.

<https://mihackeo.blogspot.com.es/>

En caso de Tomcat no ha funcionado.

CASO 4 → SSH PUERTO 22

A screenshot of a terminal window titled 'julian@pc-julian: ~'. The window has a menu bar with 'Archivo', 'Editar', 'Ver', 'Buscar', 'Terminal', and 'Ayuda'. The terminal shows a green prompt 'julian@pc-julian:~\$' followed by the command 'nc 192.168.1.37 22'. The output is 'SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.4' followed by a white cursor on a black background.

```
julian@pc-julian: ~
Archivo  Editar  Ver   Buscar  Terminal  Ayuda
julian@pc-julian:~$ nc 192.168.1.37 22
SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.4
█
```

En este último caso, también sabemos lo que corre en el SSH, el servicio OpenSSH v2 versión 7.2.

Como podemos ver en los casos, podemos consultar las cabeceras de cualquier servicio y conseguir (en muchos casos) versión y aplicación que corre en los puertos.

- **Httpprint**

Para mi la mejor herramienta para fingerprinting de aplicaciones.

Se puede bajar de aquí para cualquier plataforma: <http://www.net-square.com/httpprint.html>

Una vez descargado vamos a la ruta `./httpprint_linux_301/httpprint_301/linux` y ejecutamos:

```
./httpprint -h [ip_host] -P0 -s signatures.txt
```

y automáticamente hace el análisis mostrando el banner que ha deducido:

CASO 1

```
julian@pc-julian: ~/Descargas/Whatweb/httpprint_linux_301/httpprint_301/linux
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
julian@pc-julian:~/Descargas/Whatweb/httpprint_linux_301/httpprint_301/linux$ ./httpprint -
h 127.0.0.1 -P0 -s signatures.txt
httpprint v0.301 (beta) - web server fingerprinting tool
(c) 2003-2005 net-square solutions pvt. ltd. - see readme.txt
http://net-square.com/httpprint/
httpprint@net-square.com

Finger Printing on http://127.0.0.1:80/
Finger Printing Completed on http://127.0.0.1:80/
-----
Host: 127.0.0.1
Derived Signature:
Apache/2.4.27 (Ubuntu)
9E431BC86ED3C295811C9DC5811C9DC5050C5D32505FCFE84276E4BB811C9DC5
0D7645B5811C9DC5811C9DC5CD37187C11DDC7D7811C9DC5811C9DC52655F350
FCCC535BE2CE6923E2CE6923811C9DC5E2CE6927050C5D336ED3C295811C9DC5
6ED3C295E2CE6926811C9DC5E2CE6923E2CE69236ED3C2956ED3C295E2CE6923
E2CE69236ED3C295811C9DC5E2CE6927E2CE6923

Banner Reported: Apache/2.4.27 (Ubuntu)
Banner Deduced: Apache/2.0.x
Score: 108
Confidence: 65.06
-----
```

CASO 2

```
julian@pc-julian: ~/Descargas/Whatweb/httpprint_linux_301/httpprint_301/linux
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
julian@pc-julian:~/Descargas/Whatweb/httpprint_linux_301/httpprint_301/linux$ ./httpprint -
h 193.144.48.64 -P0 -s signatures.txt
httpprint v0.301 (beta) - web server fingerprinting tool
(c) 2003-2005 net-square solutions pvt. ltd. - see readme.txt
http://net-square.com/httpprint/
httpprint@net-square.com

Finger Printing on http://193.144.48.64:80/
Finger Printing Completed on http://193.144.48.64:80/
-----
Host: 193.144.48.64
Derived Signature:
Apache
9E431BC86ED3C295811C9DC5811C9DC5811C9DC5050C5D32505FCFE84276E4BB811C9DC5
0D7645B5811C9DC5811C9DC5CD37187C811C9DC5811C9DC5811C9DC52655F350
FCCC535BE2CE6923E2CE6923811C9DC5E2CE69272576B7696ED3C295811C9DC5
6ED3C295E2CE6923811C9DC5E2CE6923E2CE69236ED3C2956ED3C295E2CE6923
E2CE69236ED3C295811C9DC5E2CE6927E2CE6923

Banner Reported: Apache
Banner Deduced: Apache/1.3.26
Score: 89
Confidence: 53.61
-----
```

Con lo que ya sabemos la versión del apache de www.udc.es. Cabe destacar que la IP la obtuve de hacer un dig del dominio, ya que httpprint no funciona bien si se le pasa el dominio:

```
julian@pc-julian: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
julian@pc-julian:~$ dig www.udc.es

;<><> DiG 9.10.3-P4-Ubuntu <><> www.udc.es
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 3218
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;www.udc.es.                IN      A

;; ANSWER SECTION:
www.udc.es.                6878    IN      A      193.144.48.64

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53 (127.0.0.53)
;; WHEN: Fri Mar 23 13:46:58 CET 2018
;; MSG SIZE rcvd: 55

julian@pc-julian:~$
```

<https://mihackeo.blogspot.com.es/>

CASO 3

```
julian@pc-julian: ~/Descargas/Whatweb/httpprint_linux_301/httpprint_301/linux
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
julian@pc-julian:~/Descargas/Whatweb/httpprint_linux_301/httpprint_301/linux$ ./httpprint -h
127.0.0.1:8080 -P0 -s signatures.txt
httpprint v0.301 (beta) - web server fingerprinting tool
(c) 2003-2005 net-square solutions pvt. ltd. - see readme.txt
http://net-square.com/httpprint/
httpprint@net-square.com

Finger Printing on http://127.0.0.1:8080/
Finger Printing Completed on http://127.0.0.1:8080/
.....
Host: 127.0.0.1
Derived Signature:

811C9DC5E2CE6926811C9DC5811C9DC5811C9DC5D755BB19811C9DC5811C9DC5
811C9DC5811C9DC5811C9DC5811C9DC5811C9DC5811C9DC5811C9DC5811C9DC5
E2CE6926FCCC535FFCCC535F811C9DC5E2CE69272576B7696ED3C2959E431BC8
6ED3C295E2CE6926811C9DC5FCCC535FFCCC535F6ED3C295FCCC535FE2CE6923
FCCC535FFCCC535F811C9DC5E2CE6927E2CE6923

Banner Reported: -
Banner Deduced: Apache-Tomcat/4.1.29
Score: 51
Confidence: 30.72
.....
```

Para Tomcat deduce el banner, a pesar de no reportar ninguno el propio servidor, algo que muestra el potencial de la herramienta.

CASO 4

Aquí radica el verdadero problema de httpprint, **no funciona ante otro tipo de servicios** lejos de servidores web o servidores de aplicaciones web.

- **WhatWeb**

Herramienta menos potente que httpprint pero más sencilla.

Puedes bajarla aquí de GitHub: <https://github.com/urbanadventurer/WhatWeb>

```
julian@pc-julian: ~/Descargas/Whatweb/WhatWeb-master
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
julian@pc-julian:~/Descargas/Whatweb/WhatWeb-master$ sudo ./whatweb 127.0.0.1
http://127.0.0.1 [200 OK] Apache[2.4.27], Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][Apache/2.4.27 (Ubuntu)], Title[Apache2 Ubuntu Default Page: It works]
julian@pc-julian:~/Descargas/Whatweb/WhatWeb-master$
julian@pc-julian:~/Descargas/Whatweb/WhatWeb-master$ sudo ./whatweb www.udc.es
http://www.udc.es [200 OK] Apache, Cookies[SESSIONID,javax.servlet.jsp.jstl.fmt.locale.session], Country[SPAIN][ES], Google-Analytics[Universal][UA-3953244-1,UA-50881290-1], HTML5, HTTPServer[Apache/2.4.8.64], JQuery[1.7.1], Meta-Auth[Universidade da Coruña], probably OpenCms, Script[text/javascript], Title[Universidade da Coruña], X-UA-Compatible[IE=Edge]
julian@pc-julian:~/Descargas/Whatweb/WhatWeb-master$
julian@pc-julian:~/Descargas/Whatweb/WhatWeb-master$ sudo ./whatweb 127.0.0.1:8080
http://127.0.0.1:8080 [200 OK] Country[RESERVED][ZZ], HTML5, IP[127.0.0.1], Title[Apache Tomcat/8.5.3]
```

<https://mihackeo.blogspot.com.es/>

Como vemos, no es tan potente ya que no consigue la versión apache de www.udc.es pero es muy completa. Como su nombre nos indica, y al igual que httpprint, **solo vale para web**.

- **Nmap**

Nmap tiene la opción -sV para mostrar los servicios que corren en la máquina:

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
julian@pc-julian: ~
julian@pc-julian:~$ sudo nmap -sV 192.168.1.37
[sudo] password for julian:
Starting Nmap 7.60 ( https://nmap.org ) at 2018-03-24 14:31 CET
Nmap scan report for 192.168.1.37
Host is up (0.0021s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.18 ((Ubuntu))
MAC Address: 00:0B:81:A0:E7:10 (Kaparel)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 30.02 seconds
julian@pc-julian:~$
julian@pc-julian:~$
```

Como vemos, es capaz de analizar y averiguar los servicios que corren. Es el más completo de todos, y como no, el que más se usa.

- **Shodan**

Finalizamos con una extensión de navegador que tiene una gran utilidad a nivel gráfico pero ayuda bastante poco a nivel fingerprinting de aplicación.

Descarga para firefox:https://addons.mozilla.org/en-US/firefox/addon/shodan_io/

Y chrome:<https://chrome.google.com/webstore/detail/shodan/>

SHODAN | Explore | Developer Pricing | Enterprise Access | Contact Us | New to Shodan? | Login or Register

172.217.3.97 lga34s18-in-f1.1e100.net

City	Mountain View
Country	United States
Organization	Google
ISP	Google
Last Update	2018-03-20T13:57:04.008856
Hostnames	lga34s18-in-f1.1e100.net
ASN	AS15169

Ports

- 80
- 443

Services

- 80
- 443
- http

Security Contact

Contact	https://g.co/vulnz
Contact	security@google.com
Encryption	https://services.google.com/corporate/publickey.txt
Acknowledgement	https://bughunter.withgoogle.com/
Policy	https://g.co/vrpp
Hiring	https://g.co/SecurityPrivacyEngJobs

<https://mihackeo.blogspot.com.es/>

Puedes ver puertos abiertos por servidor, servicios, cabeceras, IP...

3. CONCLUSIONES

Como pudisteis ver en todo este artículo, el fingerprinting es un arma dolorosa, sobre todo cuando vayamos usar un exploit. Permitir averiguar exactamente la versión de un servicio que corremos o un sistema operativo es un gran fallo a nivel seguridad. Para ello existen ciertos trucos que nos pueden ayudar a prevenir esto:

- Cierra puertos que vengan por defecto y no uses
- Cambia TTL, MSS en la configuración de tu OS
 - Para configurar el TTL en Linux: `/proc/sys/net/ipv4/ip_default_ttl`
 - Para configurar el MSS en Linux: `sysctl net.ipv4.route.min_adv_mss`
 - ...
- Más en: <http://www.gurudelainformatica.es/2008/08/enmascarar-sistema-para-evitar-os.html>