

Python para Hackers



Python para Hackers

Contenido del curso

Capítulo 1. Introducción

Capítulo 2. Primeros pasos

Capítulo 3. Python

Capítulo 4. Hands-On

Tema de hoy

Ataques de diccionario | Web Server | Web Scraping | Fuerza bruta de Directorios Web | Fuerza bruta a formularios de autenticación (web) | Servidor/Cliente TCP | Banner Grabbing | Reconocimiento de máquinas | Nmap con Python | Packet Sniffing con Scapy | Paramiko (Cliente SSH)



Banner Grabbing

Banner Grabbing

```
savona@putor:~ — telnet
[savona@putor ~]$ telnet
telnet> open fenrir 22
Trying 10.0.0.5...
Connected to fenrir.
Escape character is '^]'.
SSH-2.0-OpenSSH 7.4
```

Es una técnica que permite **obtener información (servicios y versiones) de un equipo** dentro de una red a través de sus puertos abiertos.

Banner Grabbing

¿Cómo funciona?

Banner Grabbing Tool

DIRECCIÓN IP
10.10.10.10

LISTADO DE PUERTOS

20
21
25
80
443
8080
3306
...



Conexión	Estado	Servicio	Banner
10.10.10.10:20	Abierto	FTP	ProFTP 1.3.1
10.10.10.10:21	Abierto	FTP	ProFTP 1.3.1
10.10.10.10:25	Cerrado	SMTP	220 Welcome!
...
10.10.10.10:3306	Abierto	MYSQL	MYSQL 5.5.57



Script

BannerGrabbing.py

```
01. import socket
02.
03. def banner(ip, port):
04.     try:
05.         s = socket.socket()
06.         s.settimeout(4)
07.         s.connect((ip, int(port)))
08.         if port == 80 or port == 8080:
09.             s.send(b"HEAD / HTTP/1.1\r\n\r\n")
10.             print("Banner: " + s.recv(1024).decode())
11.         else:
12.             print("Banner: " + s.recv(1024).decode())
13.         s.close()
14.     except:
15.         pass
16.
17.
18. def scanports(ip, ports):
19.     for p in ports:
20.         try:
21.             s = socket.socket()
22.             r = s.connect_ex((ip, p))
23.             s.close()
24.
25.             if r == 0:
26.                 service = socket.getservbyport(p)
27.
28.                 print("[*] Open port: " + str(p) + " - Service: " + service)
29.                 banner(ip, p)
30.             else:
31.                 pass
32.         except:
33.             pass
34.
35.
36. ip = input("Ingresa la direccion IP a escanear: ")
37. ports = [20, 21, 22, 25, 80, 110, 113, 443, 3306, 8443]
38. scanports(ip, ports)
```

Función banner()

Línea 05 Creación del socket TCP IPV4.

Línea 06 Tiempo de espera de respuesta.

Línea 07 Conectarse a la dirección IP y puerto especificados.

Línea 08-10 Si el puerto es 80 o 8080 se envía una solicitud para obtener información del servidor HTTP.

Línea 12 Si es un puerto diferente se recibe el banner enviado por el dispositivo.

BannerGrabbing.py

```
01. import socket
02.
03. def banner(ip, port):
04.     try:
05.         s = socket.socket()
06.         s.settimeout(4)
07.         s.connect((ip, int(port)))
08.         if port == 80 or port == 8080:
09.             s.send(b"HEAD / HTTP/1.1\r\n\r\n")
10.             print("Banner: " + s.recv(1024).decode())
11.         else:
12.             print("Banner: " + s.recv(1024).decode())
13.         s.close()
14.     except:
15.         pass
16.
17.
18. def scanports(ip, ports):
19.     for p in ports:
20.         try:
21.             s = socket.socket()
22.             r = s.connect_ex((ip, p))
23.             s.close()
24.
25.             if r == 0:
26.                 service = socket.getservbyport(p)
27.
28.                 print("[*] Open port: " + str(p) + " - Service: " + service)
29.                 banner(ip, p)
30.             else:
31.                 pass
32.         except:
33.             pass
34.
35.
36. ip = input("Ingresa la direccion IP a escanear: ")
37. ports = [20, 21, 22, 25, 80, 110, 113, 443, 3306, 8443]
38. scanports(ip, ports)
```

Función scanports()

Línea 19

Por cada puerto en la lista ejecuta el código:

Línea 21

Se crea el socket.

Línea 22

Se conecta a la dirección IP y puerto indicados. A diferencia del método connect(), si no existe error devuelve el entero 0.

Línea 23

Se cierra la conexión.

Línea 25-29

Si no hubo error se obtiene el servicio que se está ejecutando en ese puerto y se llama a la función banner().

BannerGrabbing.py

```
01. import socket
02.
03. def banner(ip, port):
04.     try:
05.         s = socket.socket()
06.         s.settimeout(4)
07.         s.connect((ip, int(port)))
08.         if port == 80 or port == 8080:
09.             s.send(b"HEAD / HTTP/1.1\r\n\r\n")
10.             print("Banner: " + s.recv(1024).decode())
11.         else:
12.             print("Banner: " + s.recv(1024).decode())
13.         s.close()
14.     except:
15.         pass
16.
17.
18. def scanports(ip, ports):
19.     for p in ports:
20.         try:
21.             s = socket.socket()
22.             r = s.connect_ex((ip, p))
23.             s.close()
24.
25.             if r == 0:
26.                 service = socket.getservbyport(p)
27.
28.                 print("[*] Open port: " + str(p) + " - Service: " + service)
29.                 banner(ip, p)
30.             else:
31.                 pass
32.         except:
33.             pass
34.
35.
36. ip = input("Ingresa la direccion IP a escanear: ")
37. ports = [20, 21, 22, 25, 80, 110, 113, 443, 3306, 8443]
38. scanports(ip, ports)
```

Código principal

Línea 36 Se solicita la dirección IP a escanear.

Línea 37 Lista de puertos.

Línea 38 Se llama la función scanports().

BannerGrabbing.py

Resultado

```
Run: BannerGrabbing x
/home/kali/PycharmProjects/PyHack/venv/bin/python /home/kali/PycharmProjects/PyHack/BannerGrabbing.py
Ingresa la direccion IP a escanear: 192.168.1.115
[*] Open port: 80 - Service: http
Banner: HTTP/1.1 400 Bad Request
Date: Thu, 04 Jun 2020 18:02:17 GMT
Server: Apache/1.3.28 (Unix) mod_ssl/2.8.15 OpenSSL/0.9.7c
Connection: close
Content-Type: text/html; charset=iso-8859-1

[*] Open port: 443 - Service: https
[*] Open port: 3306 - Service: mysql
```

Reconocimiento de equipos conectados a una red

Reconocimiento de equipos



Técnica empleada **para identificar los dispositivos o equipos que están conectados a una red.**

Reconocimiento de equipos

¿Cómo funciona?

Recon Tool

SEGMENTO DE RED

192.168.1.0/24



ping 192.168.1.1
ping 192.168.1.2
ping 192.168.1.3
...
ping 192.168.1.254



Tipo de respuesta (Linux)

64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=4.81 ms

64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=4.81 ms

Destination Host Unreachable

...

Destination Host Unreachable





Script

Recon.py

```
01. from subprocess import Popen, PIPE
02.
03. for ip in range(1,254):
04.     ipAddress = "192.168.1." + str(ip)
05.     subp = Popen(['/bin/ping', '-c 1', ipAddress], //
06.                 stdout=PIPE, stdin=PIPE, stderr=PIPE)
07.     stdout, stderr = subp.communicate(input=None)
08.     if "bytes from " in str(stdout):
09.         print("IP %s is up" % (ipAddress))
```

Código principal

Línea 03

Por cada octeto en la lista del 1 al 254

Línea 04

Se concatena el octeto con la red a utilizar para formar una dirección IP.

Línea 05

Se crea un programa hijo en un nuevo proceso, el cual ejecutará el comando *ping*.

Línea 07

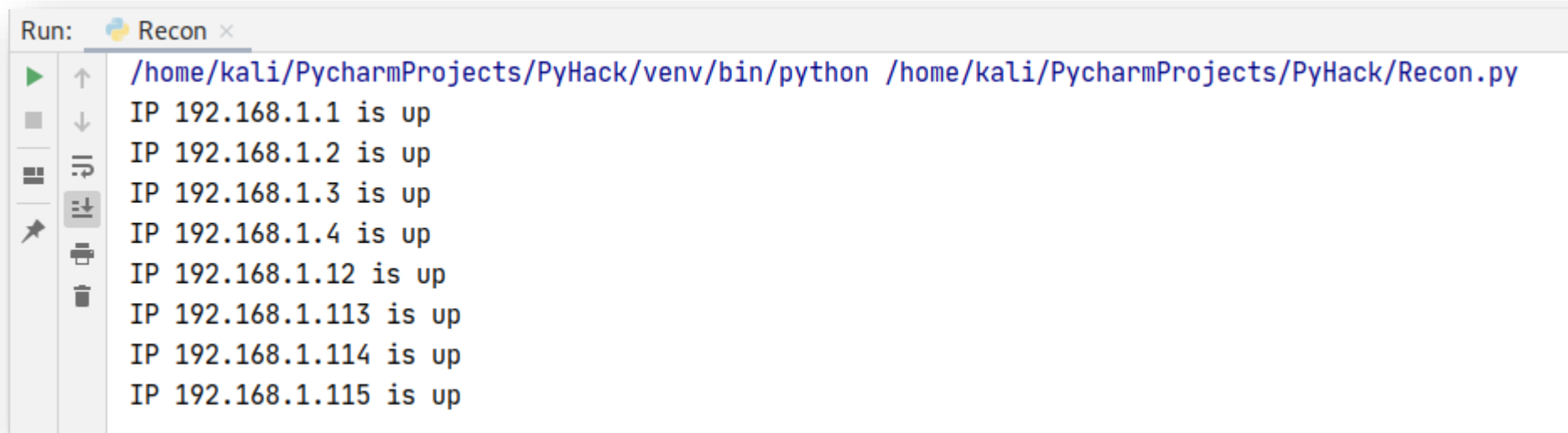
Se interactúa con el proceso. Se lee el resultado de la ejecución del comando.

Línea 08

Si en el resultado está la cadena "bytes from" quiere decir que el equipo ha respondido al ping.

Recon.py

Resultado



```
Run: Recon x
/home/kali/PycharmProjects/PyHack/venv/bin/python /home/kali/PycharmProjects/PyHack/Recon.py
IP 192.168.1.1 is up
IP 192.168.1.2 is up
IP 192.168.1.3 is up
IP 192.168.1.4 is up
IP 192.168.1.12 is up
IP 192.168.1.113 is up
IP 192.168.1.114 is up
IP 192.168.1.115 is up
```

Mostrará los dispositivos que han respondido al ping.

Utilizando *nmap* desde Python

Nmap



Herramienta que permite realizar escaneos de red para **descubrir dispositivos conectados a una red** e **identificar puertos abiertos** así como los **servicios** que se ejecutan en ellos.

Nmap

Escaneo



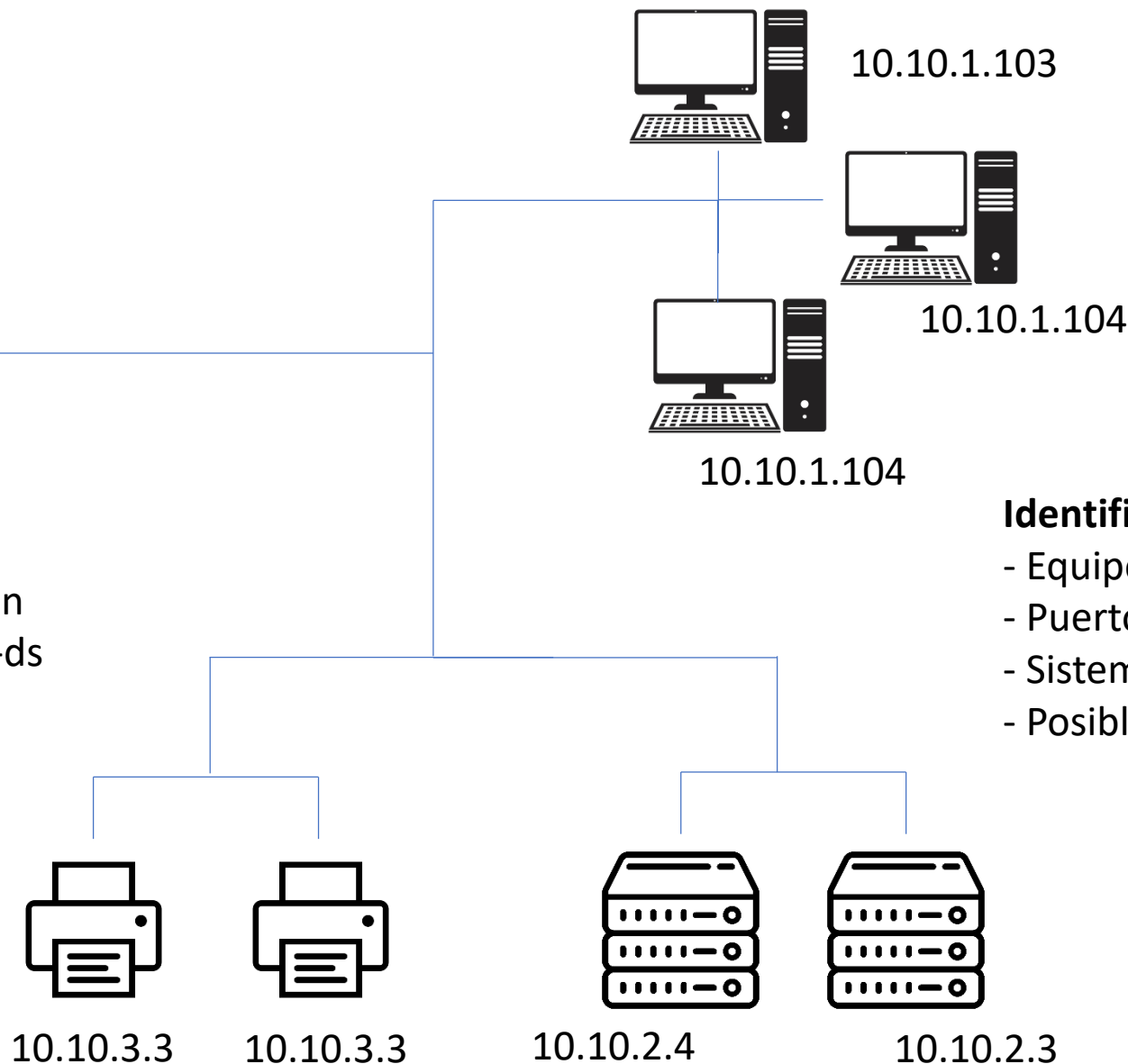
10.10.1.103 Up

Ports

139/tcp netbios-ssn
445/tcp microsoft-ds

10.10.2.4 Up

....
....
....



Identificación de:

- Equipos activos en una red
- Puertos y servicios
- Sistema Operativo
- Posibles vulnerabilidades

Script

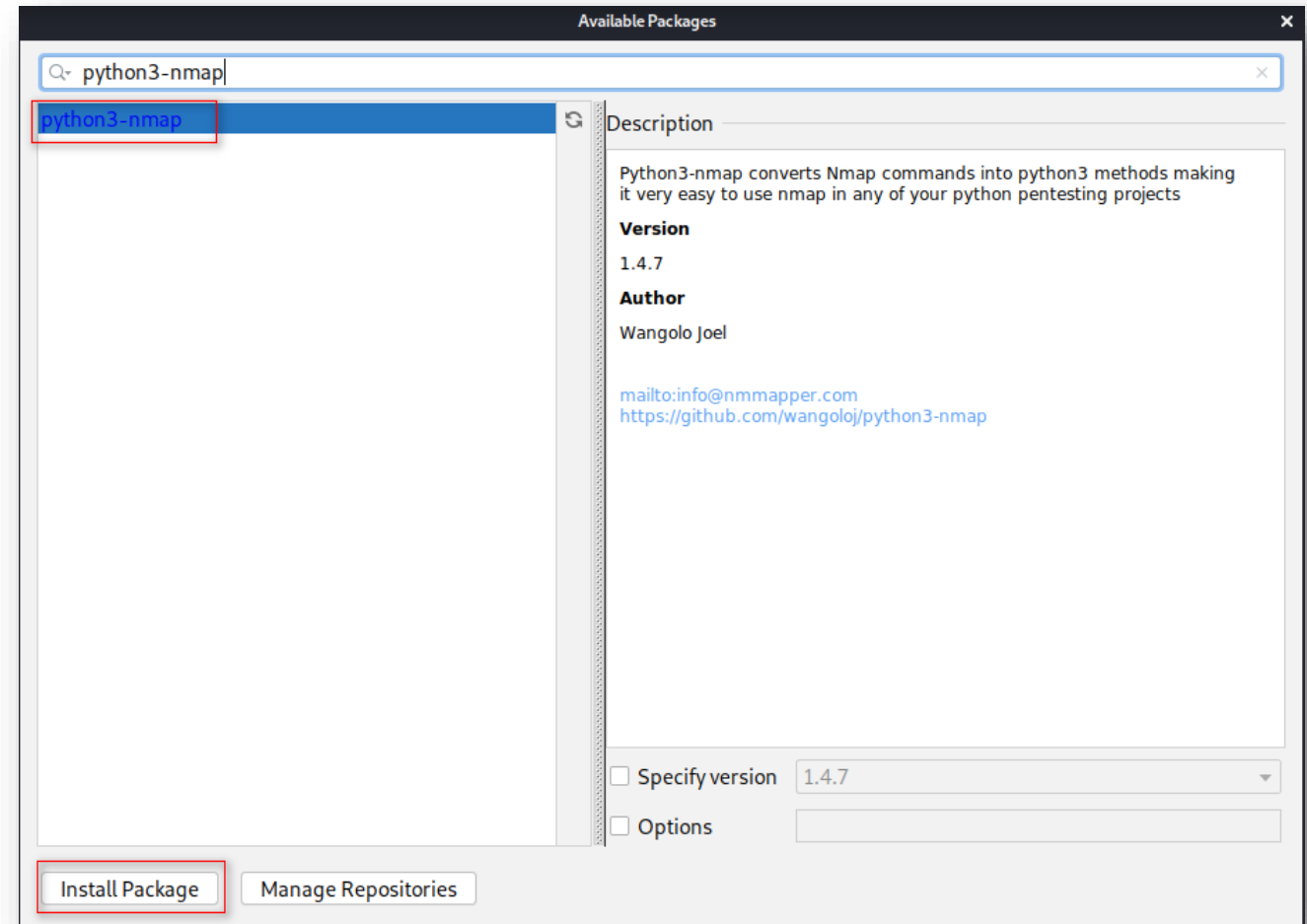
PyCharm

Instalar librería

Instalar la librería ***python3-nmap*** en PyCharm.

File > Settings > Project > Project Interpreter

- *Clic en +*
- *Buscar ***python3-nmap****
- *Instalar librería*



Nmap.py

```
01. import nmap3
02.
03. nm = nmap3.Nmap()
04. r = nm.scan_top_ports("192.168.1.115")
05. print(r)
06.
07. print("\nPuestos abiertos:")
08. for d in r.get("192.168.1.115"):
09.     if d["state"] == "open":
10.         print(d["portid"] + "/" + d["protocol"] + " ... open")
11.
12. print("\nSistema Operativo:")
13. r = nm.nmap_os_detection("192.168.1.115")
14. print(r)
15. print(r[0]["name"])
16.
17. print("\nVersiones:")
18. r = nm.nmap_version_detection("192.168.1.115")
19. print(r)
20. for d in r:
21.     s = d["port"] + "/" + d["protocol"] + " ... "
22.     if "service" in d.keys():
23.         s = s + d["service"]["name"] + " ... " + d["service"]["product"] + " " + d["service"]["version"]
24.     print(s)
25.
26. print("\nHost Discovery")
27. nmhd = nmap3.NmapHostDiscovery()
28. r = nmhd.nmap_no_portscan("", args="192.168.1.0/24")
29. print(r)
30. print("Activos: " + r["status"]["up"])
31. for h in r["hosts"]:
32.     print(h["addr"] + " ... " + h["state"])
33.
34. print("\nEspecificando argumentos")
35. nmst = nmap3.NmapScanTechniques()
36. r = nmst.nmap_tcp_scan("192.168.1.115", args="-p 21,22,80,443")
37. print(r)
38. for d in r.get("192.168.1.115"):
39.     print(d["portid"] + "/" + d["protocol"] + " ... " + d["state"] + " ... " + d["service"]["name"])
```

Código principal

Línea 01

Librería que permite utilizar algunas funciones de la herramienta *nmap*.

Línea 03

Crea un objeto de la clase *Nmap*.

Línea 04

Escanea los puertos más comunes.

Línea 13

Se obtiene el sistema operativo que está ejecutando el equipo.

Línea 18

Se identifican las versiones de los servicios en los puertos abiertos.

Nmap.py

```
01. import nmap3
02.
03. nm = nmap3.Nmap()
04. r = nm.scan_top_ports("192.168.1.115")
05. print(r)
06.
07. print("\nPuestos abiertos:")
08. for d in r.get("192.168.1.115"):
09.     if d["state"] == "open":
10.         print(d["portid"] + "/" + d["protocol"] + " ... open")
11.
12. print("\nSistema Operativo:")
13. r = nm.nmap_os_detection("192.168.1.115")
14. print(r)
15. print(r[0]["name"])
16.
17. print("\nVersiones:")
18. r = nm.nmap_version_detection("192.168.1.115")
19. print(r)
20. for d in r:
21.     s = d["port"] + "/" + d["protocol"] + " ... "
22.     if "service" in d.keys():
23.         s = s + d["service"]["name"] + " ... " + d["service"]["product"] + " " + d["service"]["version"]
24.     print(s)
25.
26. print("\nHost Discovery")
27. nmhd = nmap3.NmapHostDiscovery()
28. r = nmhd.nmap_no_portscan("", args="192.168.1.0/24")
29. print(r)
30. print("Activos: " + r["status"]["up"])
31. for h in r["hosts"]:
32.     print(h["addr"] + " ... " + h["state"])
33.
34. print("\nEspecificando argumentos")
35. nmst = nmap3.NmapScanTechniques()
36. r = nmst.nmap_tcp_scan("192.168.1.115", args="-p 21,22,80,443")
37. print(r)
38. for d in r.get("192.168.1.115"):
39.     print(d["portid"] + "/" + d["protocol"] + " ... " + d["state"] + " ... " + d["service"]["name"])
```

Línea 27

Se crea un objeto de la clase *NmapHostDiscovery*.

Línea 28

Se realiza un descubrimiento de dispositivos activos del segmento de red especificado.

Línea 35

Se crea un objeto de la clase *NmapScanTechniques*.

Línea 36

Se realiza un escaneo de puertos específicos.

Nmap.py

Equipo objetivo

En el script, cambiar la dirección IP 192.168.1.115 por **176.28.50.165**.

El servidor con dirección IP 176.28.50.165 está en internet y es para realizar pruebas, es el que se había utilizado en las pruebas de fuerza bruta.

Nmap.py

Resultado

```
Run: Nmap x
Puertos abiertos:
80/tcp ... open
443/tcp ... open

Sistema Operativo:
Linux 2.4.18 - 2.4.35 (likely embedded)

Versiones:
80/tcp ... http ... Apache httpd 1.3.28
443/tcp ...
3306/tcp ... mysql ... MySQL 4.1.7-standard

Host Discovery
Activos: 5
192.168.1.2 ... up
192.168.1.4 ... up
192.168.1.113 ... up
192.168.1.115 ... up
192.168.1.114 ... up
```

Especificando argumentos

```
21/tcp ... closed ... ftp
22/tcp ... closed ... ssh
80/tcp ... open ... http
443/tcp ... open ... https
```

Próxima actividad...

- **Evaluación:**
 - Práctica

Se notificará el detalle de la práctica en el grupo de WhatsApp.





CloudLamb

**¡Muchas gracias por su
atención!**