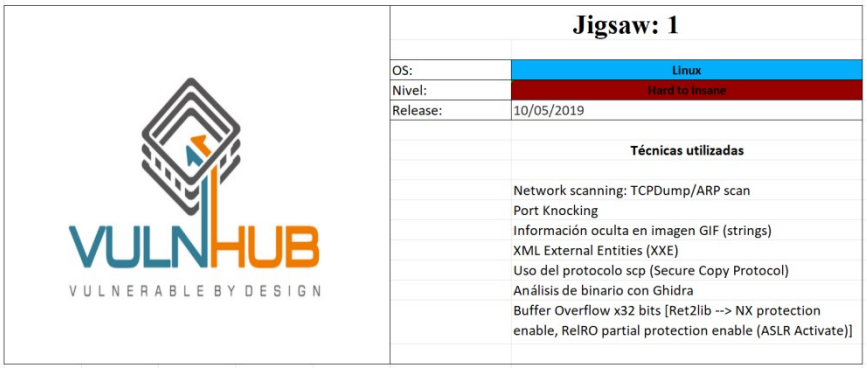


La máquina jigsaw de la plataforma de vulnhub es una máquina muy interesante de nivel difícil donde se realiza una ataque de buffer overflow utilizando la técnica Ret2lib, XML External entities entre otros, además del uso de herramientas como tcpDump.

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
```

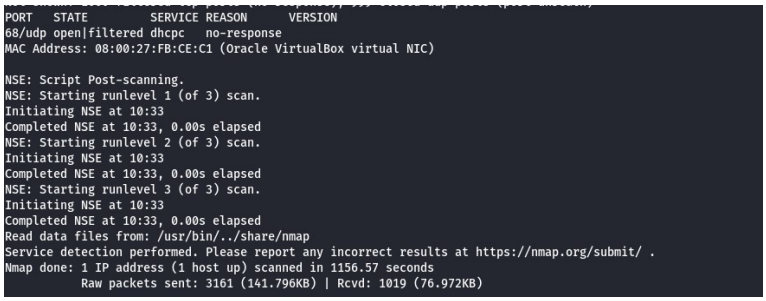
Una vez que identificada la dirección IP de la máquina objetivo, utilicé el comando **nmap -p- -sS -sC -sV --min-rate 5000 -vvv -Pn 192.168.1.12 -oN scanner_jigsaw** para descubrir los puertos abiertos y sus versiones:

- **(-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.

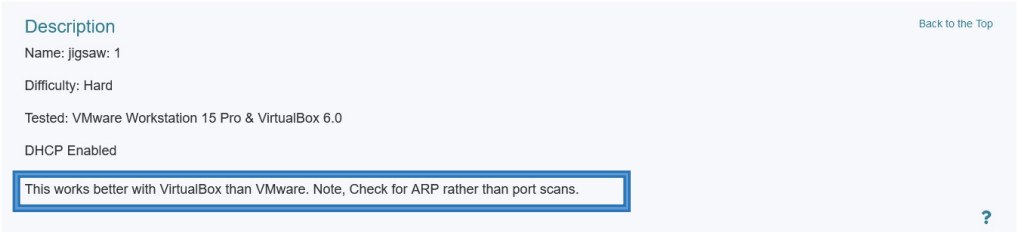
- **(-sV)**: Activa la detección de versiones. Esto es muy útil para identificar posibles vectores de ataque si la versión de algún servicio disponible es vulnerable.
- **(--min-rate 5000)**: ajusta la velocidad de envío a 5000 paquetes por segundo.
- (**-Pn**): asume que la máquina a analizar está activa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

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.



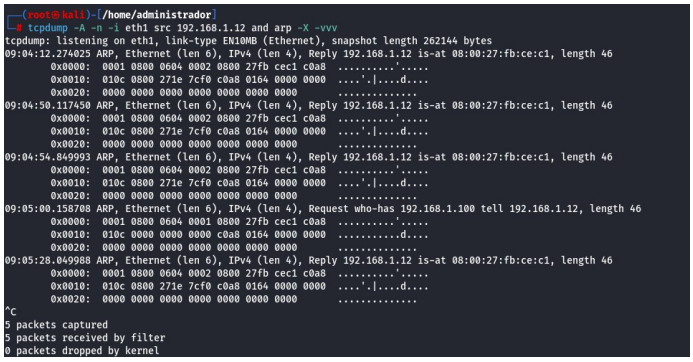
También utilicé el protocolo SCTP, además de los protocolos TCP y UDP, (Stream Control Transmission Protocol es un protocolo de red que se utiliza para transmitir múltiples flujos de datos simultáneamente entre dos puntos), pero para mi sorpresa, tampoco reveló ningún servicio o puerto abierto.

Al no encontrar nada con los protocolos TCP, UDP y SCTP, me dí 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.

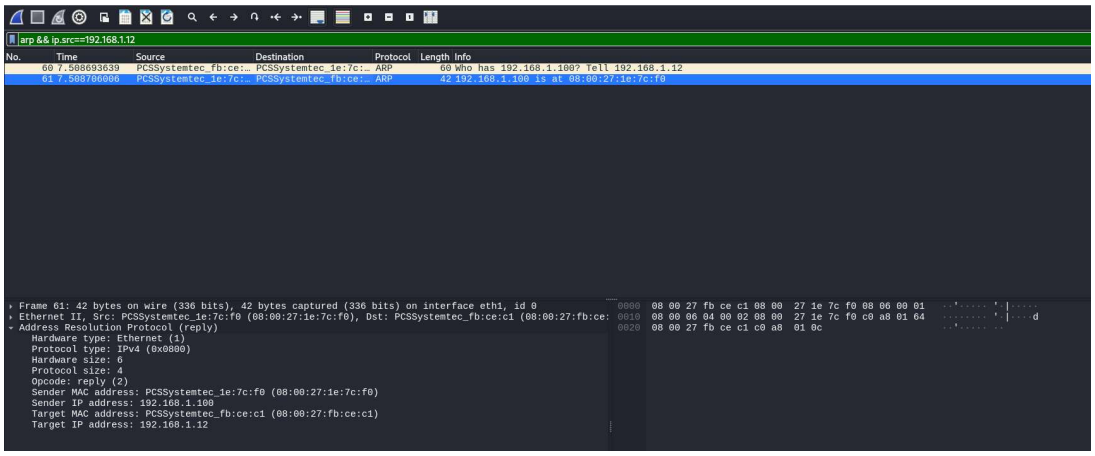


Para monitorear los paquetes ARP, utilicé el comando **tcpdump -n -i eth1 src 192.168.1.12 and arp -X -vvv** y así capturar los paquetes ARP:

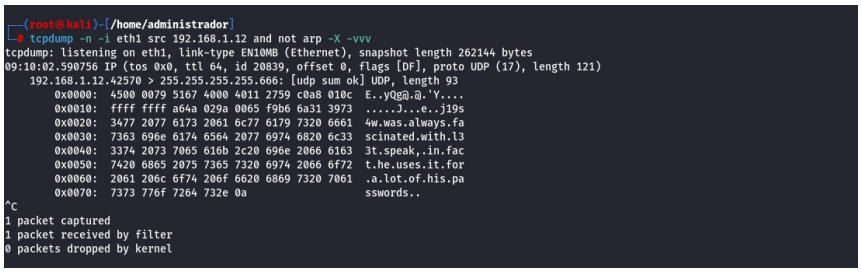
- **-n**: no resuelve nombres de dominio, lo que puede ser útil para acelerar la captura de paquetes.
- **-i eth1**: especifica la interfaz de red a utilizar para la captura de paquetes.
- **src 192.168.1.12 and arp**: filtra los paquetes ARP que provienen de la dirección IP 192.168.1.12.
- **-X**: imprime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



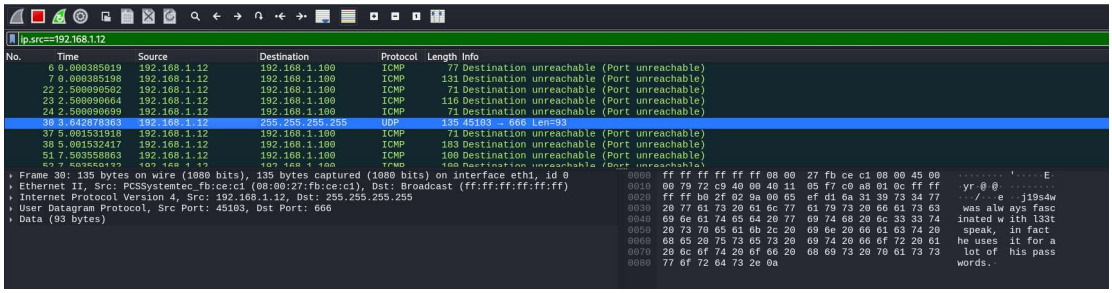
Además es posible filtrar paquete ARP utilizando Wireshark ya que puede facilitar el análisis de paquetes al utilizar un entorno gráfico. A pesar de filtrar por paquetes ARP no encontré nada interesante.



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:



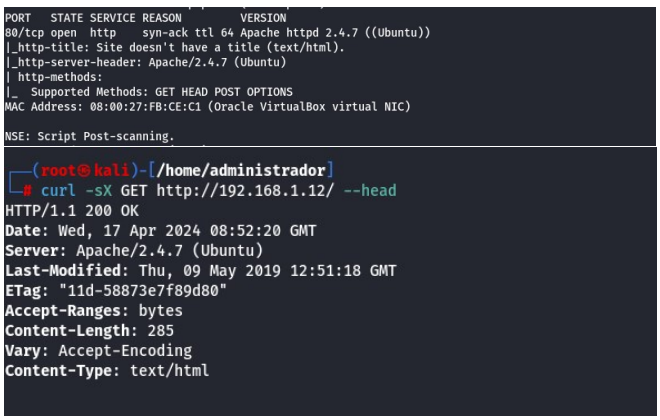
Al igual que con tcpdump, si hubiera utilizado Wireshark para analizar el tráfico de red, habría llegado al mismo resultado. El mensaje enviado indicaba que *"jigsaw está fascinado con el lenguaje leet, de hecho lo utiliza para muchas de estas contraseñas"*. El lenguaje leet es una forma de escribir en la que se reemplazan las letras con números o caracteres especiales que se parecen a las letras originales.



El mensaje anterior indica que jigsaw utiliza el lenguaje leet en sus contraseñas, así que me conecté al puerto 666 utilizando netcat y el protocolo UDP e introduje *"j19s4w"*, que es una representación en lenguaje leet de "jigsaw". Después de introducir esa contraseña obtuve una respuesta, sin embargo, la cadena de texto estaba codificada en base64.

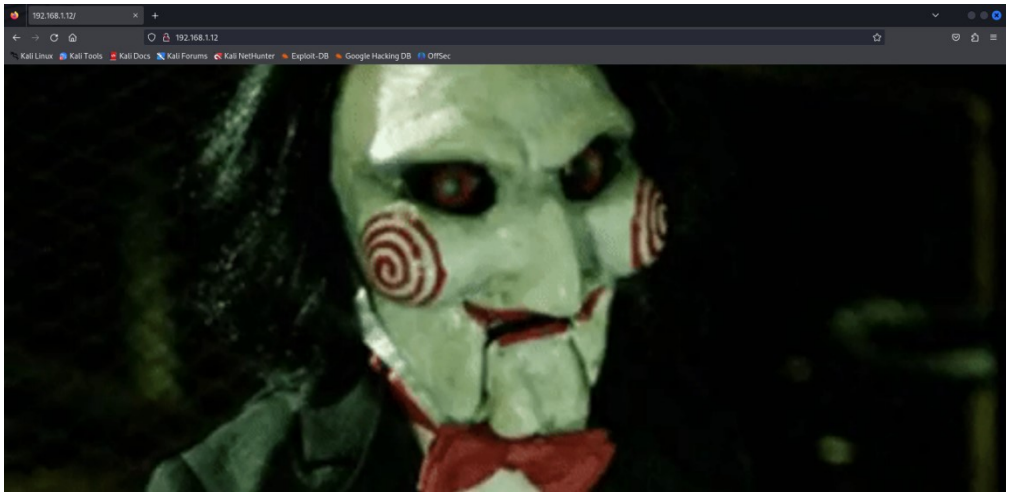
Al decodificar la cadena de texto codificada en base64 descubrí que hay puertos que se podrían desbloquear utilizando técnicas de port knoking. El port knocking es un mecanismo que permite abrir puertos en un firewall enviando una secuencia específica de intentos de conexión a diferentes puertos. Esta técnica puede aumentar la seguridad al ocultar servicios detrás de un firewall hasta que se reciba la secuencia de 'golpes' correcta.

Si el firewall que se ha configurado recibe la secuencia de conexión 5500, 6600 y 7700, éste modifica sus reglas para permitir realizar una conexión al puerto 80 (HTTP).



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 nº 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 <img src="jigsaw.gif">
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]
```

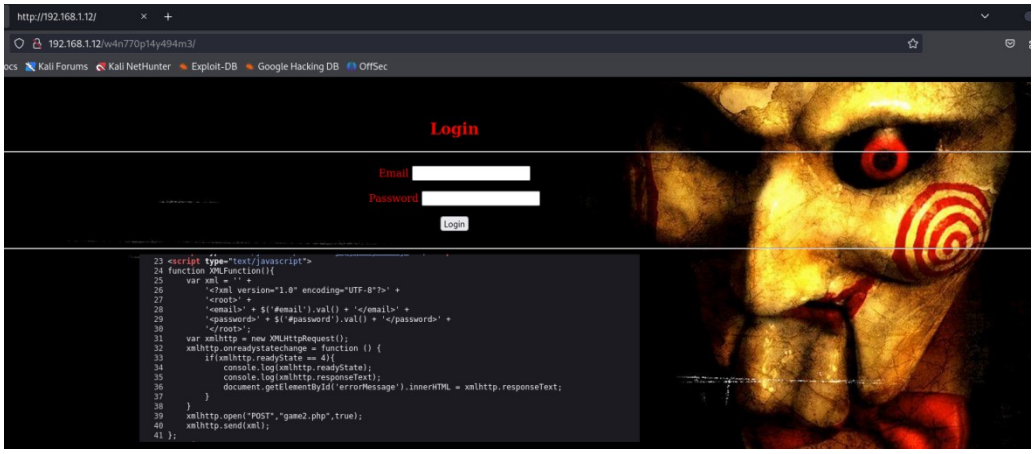
Una vez descargada analicé la imagen de una forma más exhaustiva utilizando las herramientas file para verificar que realmente se trata de una imagen y exiftool para leer los metadatos, ya que pueden contener una gran cantidad de información sobre una imagen.

```
┌──(root㉿kali)-[~administrador/Descargas]
└─# file jigsaw.gif
jigsaw.gif: GIF image data, version 89a, 500 x 302

┌──(root㉿kali)-[~administrador/Descargas]
└─# 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
```

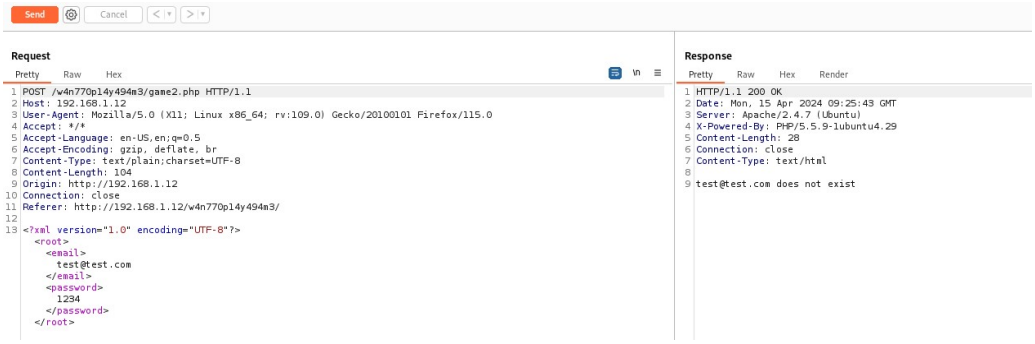
Al no encontrar ninguna información relevante en los metadatos utilicé el comando strings para encontrar las cadenas de texto legibles dentro de la imagen. La última cadena de texto era bastante diferente y, además, tenía una estructura similar a una dirección URL.

```
?1CJ
CEtDI
D\Q4EUtE
Fk4]l4Gu4
P2ud
[j9%
.(pC
je=\ZP
D:@lqh0(
*;re#(t
e\eC>
HBGt"
:/w4n770p14y494m3
```

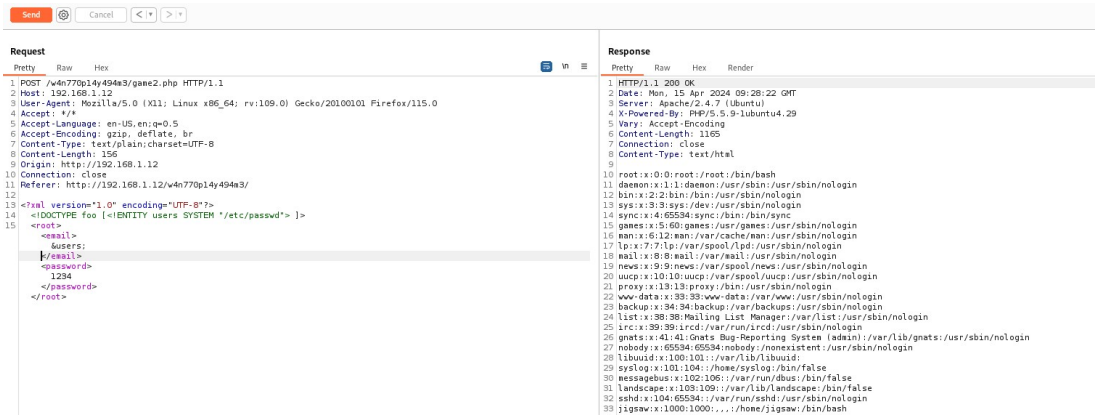
Al acceder utilizando la dirección web que descubrí, encontré un sistema de inicio de sesión donde sólo es necesario introducir un email y una contraseña. Al analizar el código fuente descubrí que la gestión de este formulario se realiza mediante código XML por lo que es posible que sea vulnerable a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



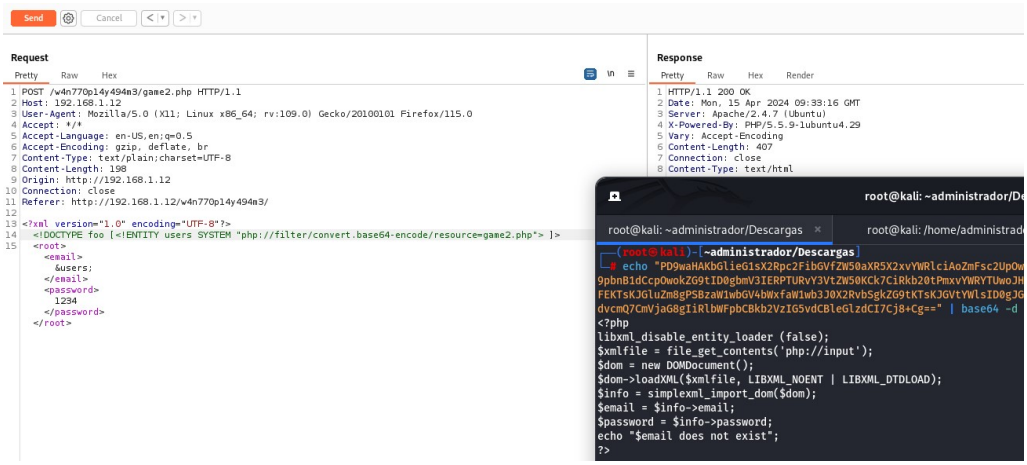
La vulnerabilidad XXE (XML External Entity) permite a un atacante manipular el procesamiento de documentos XML de una aplicación para interactuar con cualquier URI externa. Por tanto configuré el navegador web para que todo el tráfico de red lo interceptara un proxy como Burp Suite.



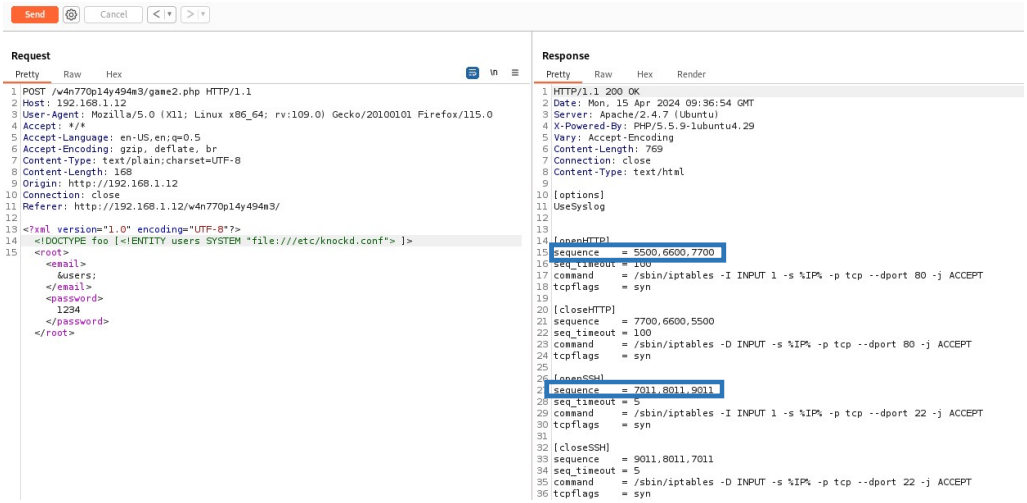
Esta vulnerabilidad permite leer archivos del servidor, así que modifiqué el código XML anterior por otro que me permitiera listar los usuarios existentes en la máquina víctima.



Este tipo de ataques permite apuntar a un archivo php y aplicar filtros de codificación en base64 para leer el código fuente de dicho archivo (PHP Wrapper inside XXE).



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

```
PORT   STATE SERVICE REASON         VERSION
22/tcp open  ssh     syn-ack ttl 64 OpenSSH 6.6.1p1 Ubuntu 2ubuntu2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 40:7f:d2:8c:69:4a:5d:a5:79:74:8c:0a:e8:30:74:05 (DSA)
| ssh-dss AAAAB3NzaC1kc3MAAACBAIVfhsL+faL8ztMvmnkROWR/jUaJGXBRiN39I+PFb+YSxMyedeotgjG+jMNM3ZjO34gdmQ9XzjLaipUoKhbkQ
AIEAgr/nuIPOjj9NI25X5IhaNv9hq+CyvUgO3oOO+iRVpS1+D0SUANeQF8pCcKbC4w8/R/SVGFIEzQo4Vx9dOvDHZCn3wrPhglVvOUntG76OQv1gCm+
Rl/nEtj2pWpI5dqVpuRa87AfIUVV4OyWQbbaHM+qr905daBPmlQzxjnccO0U02GYTK+HcVTaaHBn3opqo69LaxgAwTsISUMg7HO72z/ozmkeIcHHA=
|   2048 04:77:fb:4d:59:ef:ea:73:b7:f6:30:57:90:0c:78:81 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDA7737/vO1SG1Frw67cdnnNz5g/nTrBcpUGF6U0Y2RThfxrjhJLtoawan1R+X5D0KXLMKi4Cgy9
lZUT6lTG6soP9a5X3j1CdUIV8ti155ngphPt4rYufZ12arUJIPOpVtFg3zwJaQ8w4O/WhbHTK22FXebyQkqw1/z2JS3XAv2lBVCo6C49iqUQDEHJOaG
|   256 82:dc:9d:9e:a8:a0:ba:36:a9:6f:e4:9d:5e:96:fa:ae (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBCqqjfiTsg1fr8kn5DiPgxQfr8q6B23u6zu6+gBnu
|   256 e1:16:e0:93:20:33:be:ff:97:4e:b5:79:08:f5:41:e7 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIBgXqmHHgehMTMzrLeo8EQINslD8Dy/V9CICqlJW5Xjo
80/tcp open  http    syn-ack ttl 64 Apache httpd 2.4.7 ((Ubuntu))
|_http-title: Site doesn't have a title (text/html).
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.7 (Ubuntu)
MAC Address: 08:00:27:FB:CE:C1 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

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:

```
┌──(root㉿kali)-[~administrador/Descargas]
└─# ssh jigsaw@192.168.1.12
The authenticity of host '192.168.1.12 (192.168.1.12)' can't be established.
ED25519 key fingerprint is SHA256:a84xq+c6oF0z2v+gpwu8+hgR0wy18zkF6k15PbVPhsg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.12' (ED25519) to the list of known hosts.
jigsaw@192.168.1.12's password:
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 4.4.0-146-generic i686)

 * Documentation:  https://help.ubuntu.com/

  System information as of Mon Apr 15 01:37:06 CDT 2024

  System load: 0.0                Memory usage: 4%   Processes:       85
  Usage of /:  14.8% of 11.84GB   Swap usage:   0%   Users logged in: 0

  Graph this data and manage this system at:
    https://landscape.canonical.com/

jigsaw@jigsaw:~$ id
uid=1000(jigsaw) gid=1000(jigsaw) groups=1000(jigsaw)
jigsaw@jigsaw:~$ ls -l
total 4
-rw------- 1 jigsaw jigsaw 103 May 10  2019 y0ud1dw3118u7175n070v32.txt
jigsaw@jigsaw:~$ cat y0ud1dw3118u7175n070v32.txt

flag2{a69ef5c0fa50b933f05a5878a9cbbb54}
Hack or fail. Make your choice... Now comes your final test.
```

Después de iniciar sesión investigué los posibles archivo 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-sr-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/dmcrypt-get-device
-rwsr-xr-x 1 root root 9804 Feb 11  2014 /usr/lib/policykit-1/polkit-agent-helper-1
-rwsr-xr-- 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-sr-x 1 libuuid libuuid 17996 Jun  3  2014 /usr/sbin/uuidd
-rwsr-xr-- 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/game3
-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-sr-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/dmcrypt-get-device
-rwsr-xr-x 1 root root 9804 Feb 11  2014 /usr/lib/policykit-1/polkit-agent-helper-1
-rwsr-xr-- 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-sr-x 1 libuuid libuuid 17996 Jun  3  2014 /usr/sbin/uuidd
-rwsr-xr-- 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/game3
-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.

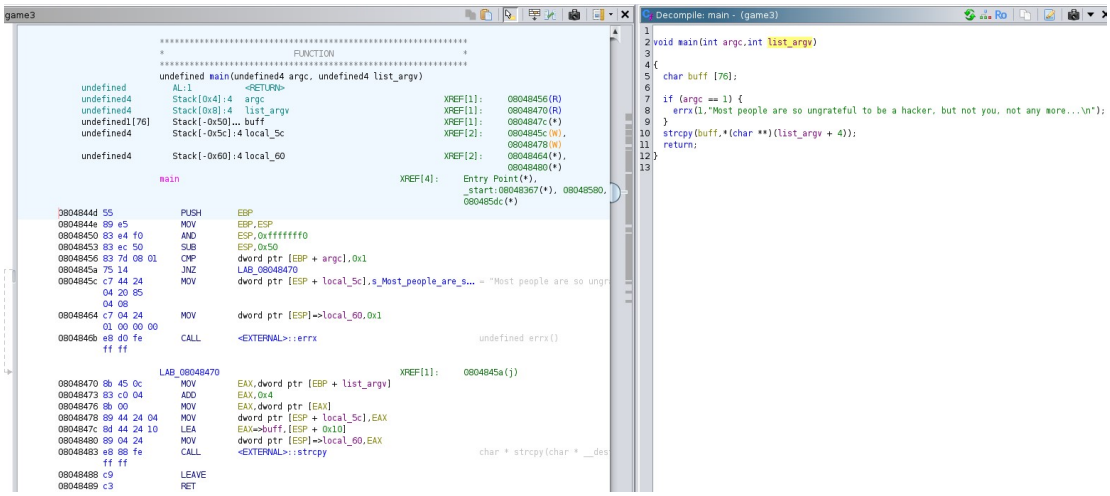
```
jigsaw@jigsaw:~$ shred -n 35 -fuvz LinEnum.sh
shred: LinEnum.sh: pass 1/36 (random)...
shred: LinEnum.sh: pass 2/36 (555555)...
shred: LinEnum.sh: pass 3/36 (924924)...
shred: LinEnum.sh: pass 4/36 (999999)...
shred: LinEnum.sh: pass 5/36 (aaaaaa)...
shred: LinEnum.sh: pass 6/36 (5b6db6)...
shred: LinEnum.sh: pass 7/36 (eeeeee)...
shred: LinEnum.sh: pass 8/36 (dddddd)...
shred: LinEnum.sh: pass 9/36 (6db6db)...
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 (7fffff)...
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 000000000
shred: 000000000: renamed to 00000000
shred: 00000000: renamed to 0000000
shred: 0000000: renamed to 000000
```

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.

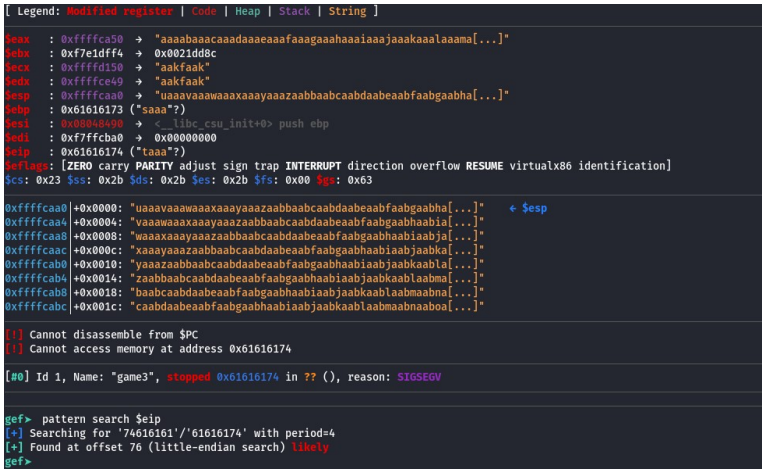
```
┌──(root㉿kali)-[/home/administrador/Descargas]
└─# scp jigsaw@192.168.1.12:/bin/game3 /home/administrador/Descargas
jigsaw@192.168.1.12's password:
game3

┌──(root㉿kali)-[~administrador/Descargas]
└─# readelf -h game3
Encabezado ELF:
  Mágico:  7f 45 4c 46 01 01 01 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)-[~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    DirFísica  TamFich TamMem  Opt Alin
  PHDR           0x000034 0x08048034 0x08048034 0x00120 0x00120 R E 0x4
  INTERP         0x000154 0x08048154 0x08048154 0x00013 0x00013 R   0x1
      [Requesting program interpreter: /lib/ld-linux.so.2]
  LOAD           0x000000 0x08048000 0x08048000 0x00648 0x00648 R E 0x1000
  LOAD           0x000f08 0x08049f08 0x08049f08 0x0011c 0x00120 RW  0x1000
  DYNAMIC        0x000f14 0x08049f14 0x08049f14 0x000e8 0x000e8 RW  0x4
  NOTE           0x000168 0x08048168 0x08048168 0x00044 0x00044 R   0x4
  GNU_EH_FRAME   0x00056c 0x0804856c 0x0804856c 0x0002c 0x0002c R   0x4
  GNU_STACK      0x000000 0x00000000 0x00000000 0x00000 0x00000 RW  0x10
  GNU_RELRO      0x000f08 0x08049f08 0x08049f08 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.

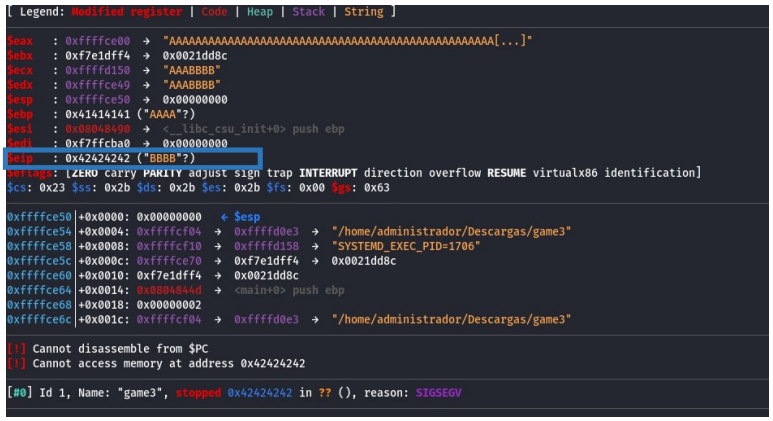


Antes de continuar es necesario conocer el tipo de protecciones que tiene activado. En concreto la protección NX-No eXecute- (marca las regiones de la memoria como no ejecutables) está activada y la protección RelRO -Relocation Read-Only- (hace que ciertas tablas de reubicación sean de solo lectura, en concreto, previene la sobreescritura 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.

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



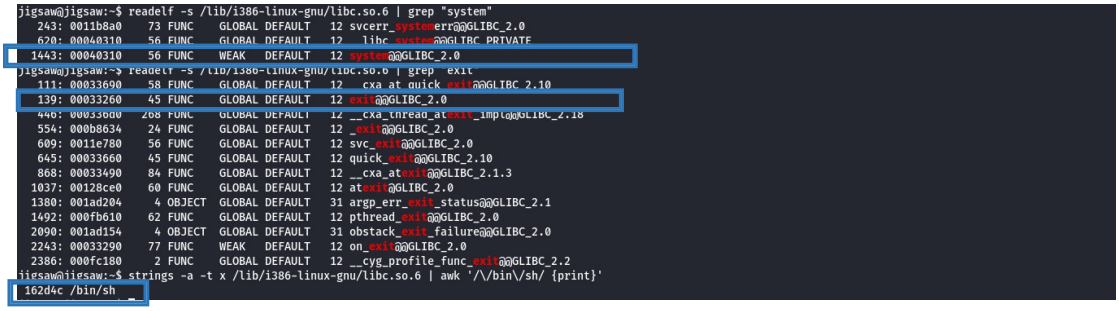
Los caracteres introducidos sobrescriben el registro $eip. Sabiendo esto utilicé la función pattern search de gdb para determinar el offset exacto para controlar dicho registro.



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:



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

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

```
jigsaw@jigsaw:~$ python3 exploit.py
game3: ../iconv/skeleton.c:736: __gconv_transform_utf8_internal: Assertion `outbuf == outerr' failed.
****": Assertion `next->fd_nextsize->bk_nextsize == next' failed.
game3: ../posix/glob.c:1675: •Ö!: Assertion `old == &init_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 Knoking 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.cDnW58
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 = "ZmxhZzF7MzAzNGNjMjkyN2I1OWUwYjIwNjk2MjQxZjE0ZDU3M2V9CllvdSBjb21wbGV0ZWQgeW91ciBmaXJzdCB0ZXN0LiBOb3cga25vY2sgdGhlc2UgbnVtYmVycyB0byBmaW5kIHdoYXQgeW91IHNlZWsuIDU1MDAgNjYwMCA3NzAw"

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)
```