Cyberhack2077

Cybersecurity challenge

Writeup of challenges

Team:
John Keys Del Pwn
(alseTS)

December 3, 2023

Contents

1 Introducción 1

2 Initial foothold 2

3 First flag and initial access to the machine 5

4 Second flag and *pivoting* from Faraday 10

5 Third flag and *pivoting* to user kiwi 12

6 Fourth flag and *pivoting* to user Adam Smasher 21

7 Fifth flag and elevation of root privileges 24

8 Sixth and last flag of user David Martínez 26

9 Conclusions 27

1 Introducción

To contextualize the proposed challenge, we will first introduce what the challenge consisted of and what was the initial information available. The description of the challenge is as follows:

> 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"

The objectives of the challenge were as follows:

- Obtain maximum privileges in the system
- Collect as many flags as possible

In addition, two different resources are made available to the participant: a participant's guide on how to send the flags and a wordlist which includes a dictionary to be used for the challenges.

2 Initial foothold

Once the challenge has been introduced, an IP address is given to each participating machine. In this case, we will work on IP 13.38.118.229. Both in the description of the challenge and during the introductory talk of the event, participants are told that the only open ports on the machines will be 22 (SSH) and 1337. Since, for the moment, we do not have credentials to log in via SSH, we will connect to port 1337 of the IP address provided. First we tried to connect via HTTP without success, so we tried to access via TCP protocol using netcat (nc). In doing so, we found a console with which we can interact:

Figure 1: Initial connection to the machine provided.

As we can see, we have a command *actions* that allows us to visualize

which actions we can carry out in the service. The first one that catches our attention is contacts. In this menu different messages are displayed, which will be the necessary clues to continue with the challenge. Specifically, Lucy's message tells us that she has left a back door on Faraday's device and that we will be able to access it with the necessary configuration. If we look at the available actions, the only one that can be configured by the user is the *gear* option. To find out what options we have, we launch the command *deck*:

Figure 2: Option deck from the list of configurable actions and components.

As we can see, we have 3 options available for each of the 3 elements belonging to the chip (which can be displayed with the command *chip*). If we go back over the messages shown in Figure 1, we realize that Maine's message has something strange: it sets the rendezvous time to 31:20, a non-existent time and that could be a clue to find out the correct configuration. To solve this part, a Python script was developed using the pwntools library (solver1.py) that can be found in this Github repository. Once the script execution is finished, we get correct configuration:

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

If we look closely, they coincide with the digits of the proposed meeting time but in reverse order. If we establish this configuration, we can see that

when we execute the command *actions* again, a new option appears that allows us to interact with the machine that hosts the service: *net_debug*.

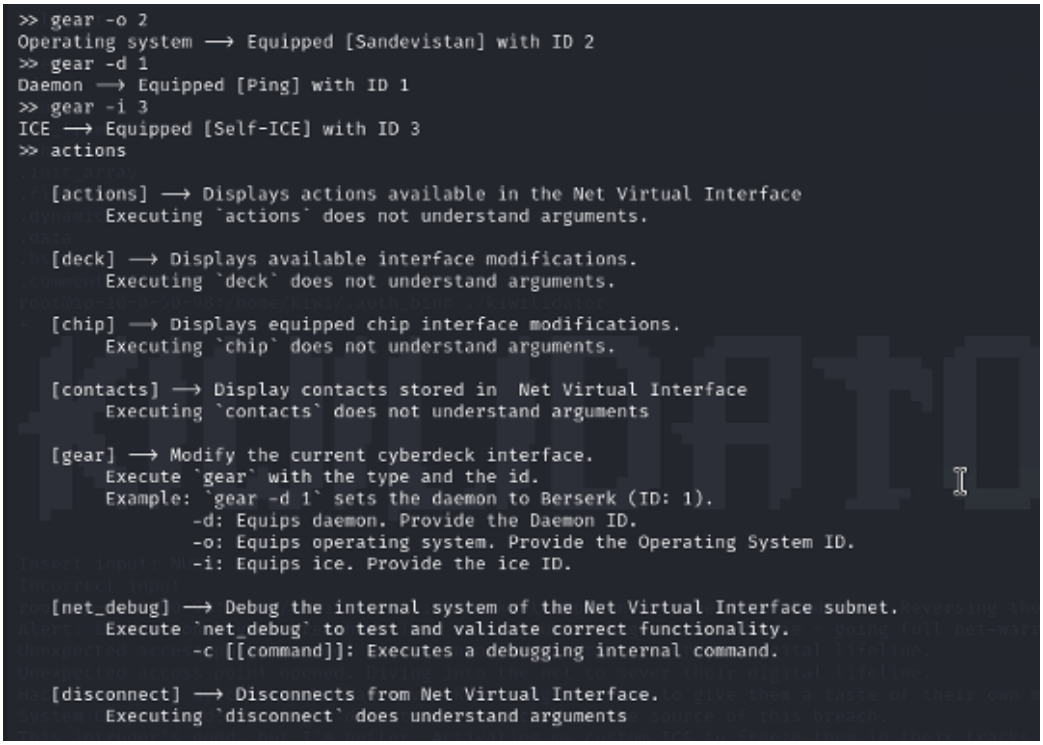


Figure 3: Correct configuration of *gear* and new action available.

3 First flag and initial access to the machine

Using the command *net_debug -c ls* we see that in the current directory we have 3 elements: an image (inmunosupresor.jpg), an audio file (intercepted.wav) and a text file (flag.txt). Executing *net_debug -c cat flag.txt* we obtain the first flag [1]. Once the flag is retrieved, we get the files we found in the directory. The way we chose to exfiltrate this content was to take advantage of the *net_debug* command itself and obtain the files encoded in base64:

[1] No screenshot of this flag is available because the service stopped working and the corresponding screenshots could not be taken in time

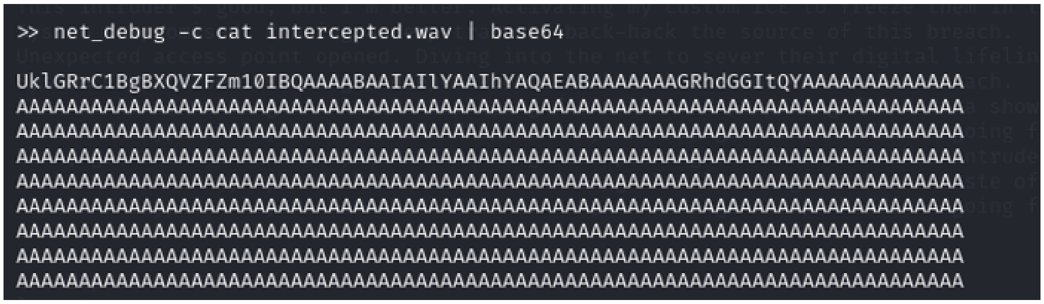


Figure 4: Exfiltration of files.

The first thing we will check is whether the image contains any information hidden through steganography. To do this, we use the Stegseek tool, which performs a dictionary attack based on the Steghide tool. As shown in Figure 5, a hidden file called "partone" was included within the JPG.

Figure 5: Executing the Stegseek tool on the image inmunosupresor.jpg

If we look at the content of the file, we see that it is base64 encoded.

Figure 6: Contents of the "partone" file.

If we proceed to decode this information, we see that it is an OpenSSH key:

```
┌──(stesla㉿kali)-[~/Descargas/NUWECHALL]
└─$ cat inmunosupresor.jpg.out | base64 -d
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAn6y/kJsKxPH8RzZyZX7hrsBiO74ZzYOAlHjFisvTbh/d+hWQW78l
91SOPkS7EaMCAnXNrpWzfOVMF3S2lfRVM2kFCq0trXEmm4bXmS6Nk/usR+Vpcl3iHh6g7H
jsDCzJTbG3RRHvLhkB/FOCZeiDOEEDmab6lnYh9scI6dIpe4EpEyogGT0CQj4RJOKtp51u
Pz8QNEH4UL0WBtsD8kTYFRNym1TdzZDth3nYDRiTYMTIvhqgnC4PmBzrbMvSYCkf9eaxqo
JUs79C8eyw5q+BD7OFhpYBH0cyjLef0NFaSFfPwuHyhVgI/htdb3uu35j6rJml0+f+1tfc
FLgPPbqx0oBjKJ/o8XmMlIkSwY0MfH4zboLZVcwE2HT/izOu/zKxuLIkofjs30EAOokWmW
YH41n5DmhznSNiDzBcKmWEBwKJLQEnblcankih+SibjcLs7SEXmbtNHnwrYZeOLv67fYTx
YW7a1OKNVffg4m3fuwrihx8UXLOTktNSh0+8Rm81AAAFkMYoXE7GKFxOAAAAB3NzaC1yc2
EAAAGBAJ+sv5CbCsTx/Ec2cmV+4a7AYju+Gc2DgJR4xYrL024f3foVkFu/JfdUjj5EuxGj
AgJ1za6Vs3zlTBd0tpX0VTNpBQqtLa1xJpuG15kujZP7rEflaXJd4h4eoOx47AwsyU2xt0
UR7y4ZAfxTgmXogzhBA5mm+pZ2IfbHCOnSKXuBKRMqIBk9AkI+ESTiraedbj8/EDRB+FC9
FgbbA/JE2BUTcptU3c2Q7Yd52A0Yk2DEyL4aoJwuD5gc62zL0mApH/XmsaqCVLO/QvHssO
avgQ+zhYaWAR9HMoy3n9DRWkhXz8Lh8oVYCP4bXW97rt+Y+qyZpdPn/tbX3BS4Dz26sdKA
Yyif6PF5jJSJEsGNDHx+M26C2VXMBNh0/4szrv8ysbiyJKH47N9BADqJFplmB+NZ+Q5oc5
0jYg8wXCplhAcCiS0BJ25XGp5Iofkom43C7O0hF5m7TR58K2GXji7+u32E8WFu2tTijVX3
4OJt37sK4ocfFFyzk5LTUodPvEZvNQAAAAMBAAEAAAGAAYKaQI1e1t8CXPq5rpZlHwh9xV
qLwZhWonuD7sdKZXZoBzdQqWAk7/JnnVTRah4fD5PlUUXkL9sL/bAiZp0BeJ4r0zPbxJIB
5KMBWKxf4BWXcF9N6ywnzn4eqoM4RE3gzNsl7xbM1XmcvbIH1TE+Rc1uRx6VAdBLNkYf95
9/ufB49CVIBBpsx4tGZpFoRWIP9V7XfI8KgK+md4PMb+xRGN3uuGO0l1RMEtBxsn9YLChK
97/7uI5s96oqWy86+vgt6+E++n6Tb48TG0477HNuoOqxWXlAbCDCfCdTijI9Fk8lxNLotT
w/QtvDom3vlP6d7yTKnQfzWRh7JPpxdVoSrKH3AyggfGS4cD5xPEGxcp07xoZmyY2ZI7K9
sDWFXwglU4tqQGTGtSBOFqKy4yDMp4WUC0fkXSXZPHordC9WWUr4tWbdAW9Hm2fhw5McyF
dUBpUn1FziEwiU5iJ4u4IoK2p88uTugsotxtBFe0mitgXZsesvXQ8s1uAUk/iY6r/PAAAA
wA4cJzCypABRvnZxtRT/74+W2IjrHy7ZarpTxjX8jsIKRdT2ZZMQdYznJVGlVfP31D/FEe
blBNUiPooogK2SwVG41wrypv/LGbcztirI+rctVJPxZoXSLQCtL+kTWNDon85UOTU86ell
q5OfIpmpEeglWVNJx8H2UUmz4XJ8xeV8PCTj2PCZ88JZ8IbsMTk+nHz2adcq6dDwrFiOOI
RS87+aDwAeHhdzuf2oLb/zoGDYH3jTINp12AmMQy4E8JcDdAAAAMEAyou2Xsi2o6ouvPW6
q6Dzj9lPAAAvGqoA53P9/YbWnjTAhF5Xl0losPiNz8UicPi15mfzaami8BxPxNMt85MZif
petLW6VxevGUH2mMqsASvKwPgreR0IoacFzD+AAqxbU0eV9rG/49yOSS8XEzGH2oCE6hbo
sUpTWjfV8w0s8T6cjw4/2fCwLtwbVjSQiOdBVH6wbm6RWVDeLlqJPEdxIvzAWx9QXiSTVW
DfciDms3SFm9jyJjPQyj1ZpVLMES7TAAAAwQDJ0JyMIgBN9hpzmiU85/jOz9iz5/v9GsW/
OW7ySjDV42xSZym0UpGG4PAsCj+Apc4QB862AVoGDfui36y3CjDx3uSrVr7xcWwxVCDlQs
8K6D2vvX0ThieqJ18GHMNxeDdFyi96vMYuiDhIcqHFzQ+hf/CnrvY4PJKrB0iZdBEpu6nU
b+ExS+WTFdxJPhmzPfQvlcYEY3mnTIIdQ660iMC4kLbSLr7TP23bH5doRHAMyyOJXsxGG9
pxMpe7/LWN9NcAAAAVZmFyYWRhe
-----END OPENSSH PRIVATE KEY-----
```

Figure 7: Content of decoded “partone”.

If we try to perform some action such as recovering the public key from the private one with OpenSSL, we see that we get an error from Libcrypto. This is because, as the name of the hidden file itself indicates, we only have the first part of the SSH key, so we must find the second part for it to work correctly. If we open the file *intercepted.wav* with Sonic-Visualiser and add the spectrogram layer, we will find the final part of the key: UBjeWJlcmhh-

hhY2syMDc3AQIDBAUG

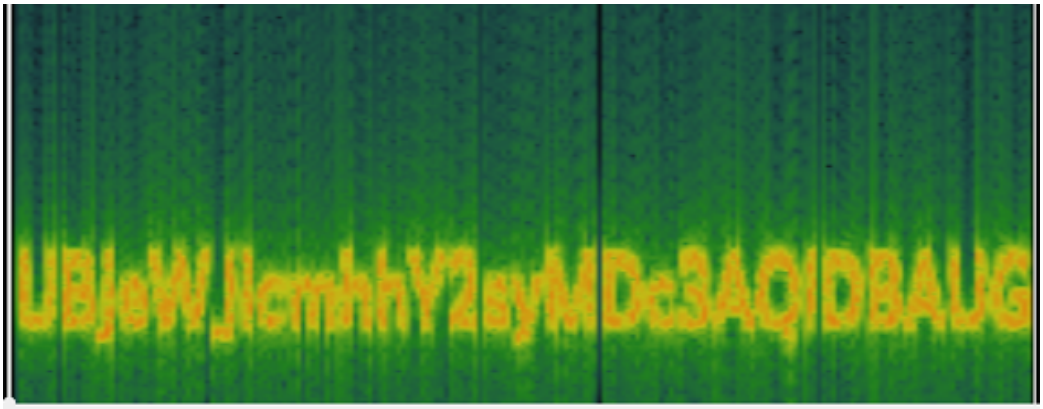


Figure 8: Spectogram of the file *intercepted.wav*.

With this information, we now have our OpenSSH key correctly constructed. If we look again at the messages shown in Figure 1, we remember that Lucy was talking about a backdoor on Faraday's device, so it is very likely that the recovered OpenSSH key belongs to this user. If we try to connect via SSH using this identity we will see that we are right:

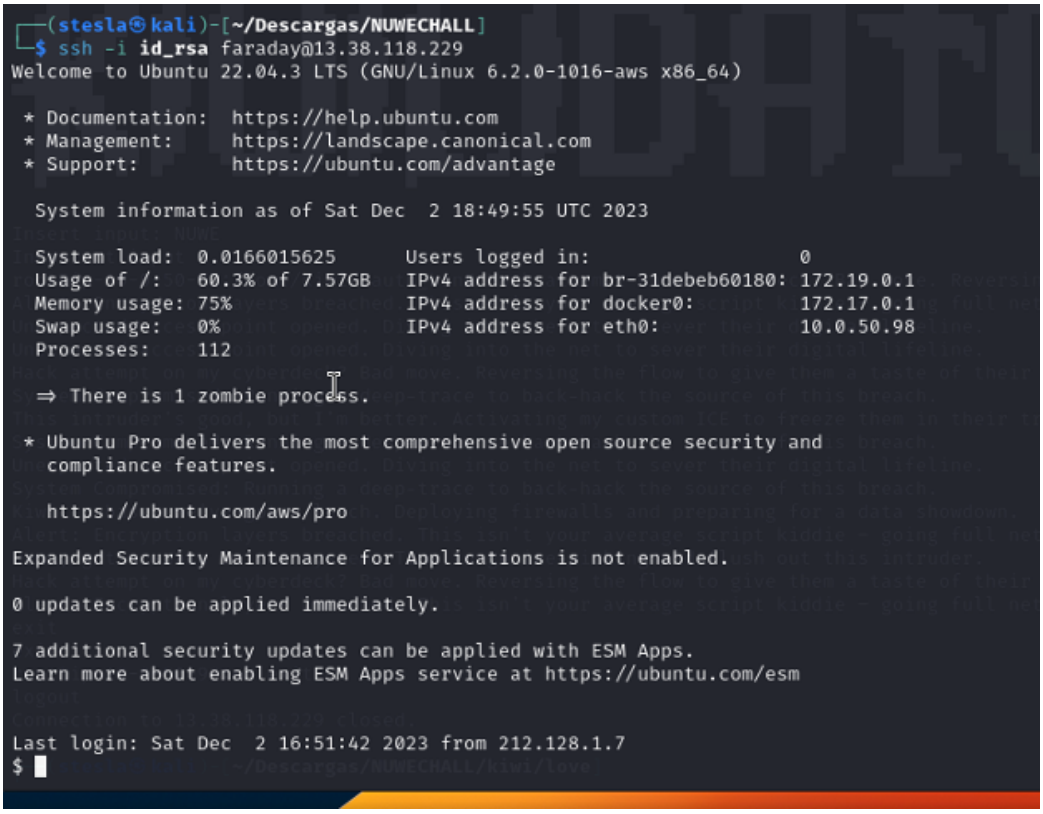


Figure 9: Successful login to the Faraday machine.

4 Second flag and *pivoting* from Faraday

Once logged into the machine, the first thing we will do is download the user's home to our local machine to keep the evidence in case we lose access to the machine for any reason. Once this is done, we analyze the files found without much success. We then go back to the machine and analyze which ports are open to find potential vulnerable or interesting services.

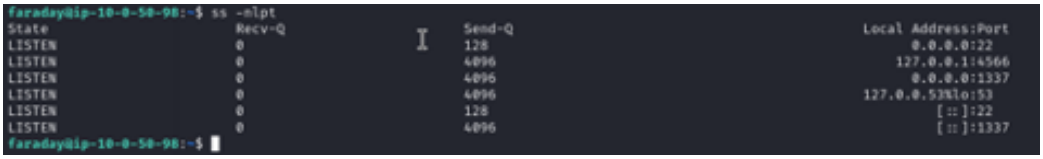


Figure 10: Analysis of open ports on the local machine.

As we can see, there are two interesting ports: 1337 and 4566. The 1337 corresponds to the chip service, so we discard this way. Investigating a bit, we see that port 4566 is a common port for Localstack, an Amazon Web Services application that allows us to test our Cloud applications on a local machine. We also see that the awslocal tool is installed, so we can list instances. We will do just this with the s3 instances:

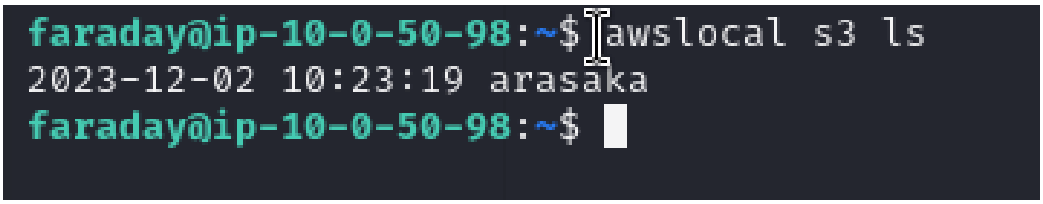


Figure 11: List of S3 instances

As we can see, there is an S3 instance called arasaka, a name mentioned in the challenge description (Section 1). If we list the contents of this instance, we see something interesting: a file *note.txt*.

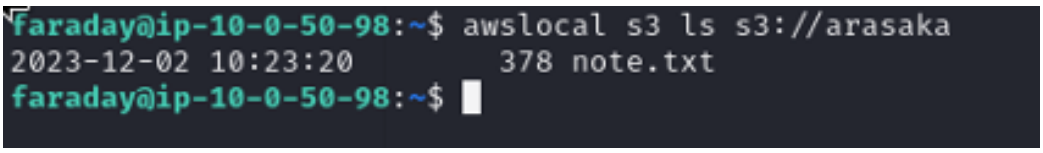


Figure 12: List of the contents of the arasaka instance.

We download it to our local machine and open its content, observing that we have a link to a memory dump.

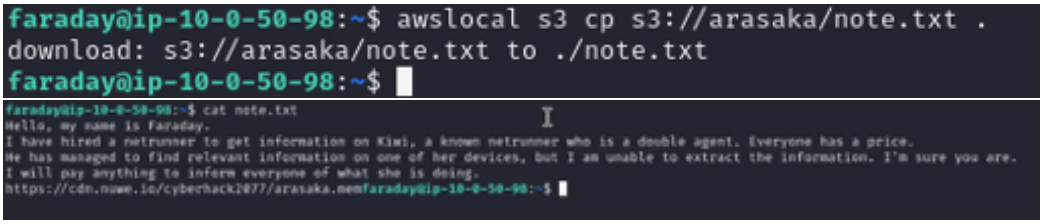


Figure 13: Copy and content of the note.

Once the memory dump is downloaded, we will obtain the following flag using an "unorthodox" method, but which is just as valid as using memory analyzers: using the strings command. Strictly speaking, the correct methodology to follow would have been to dump the memory of the Notepad.exe process that contained the flag.txt file. However, using strings speeds up the process:

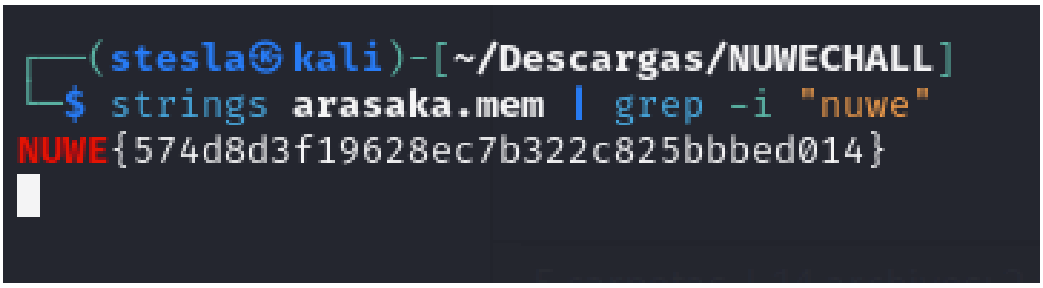
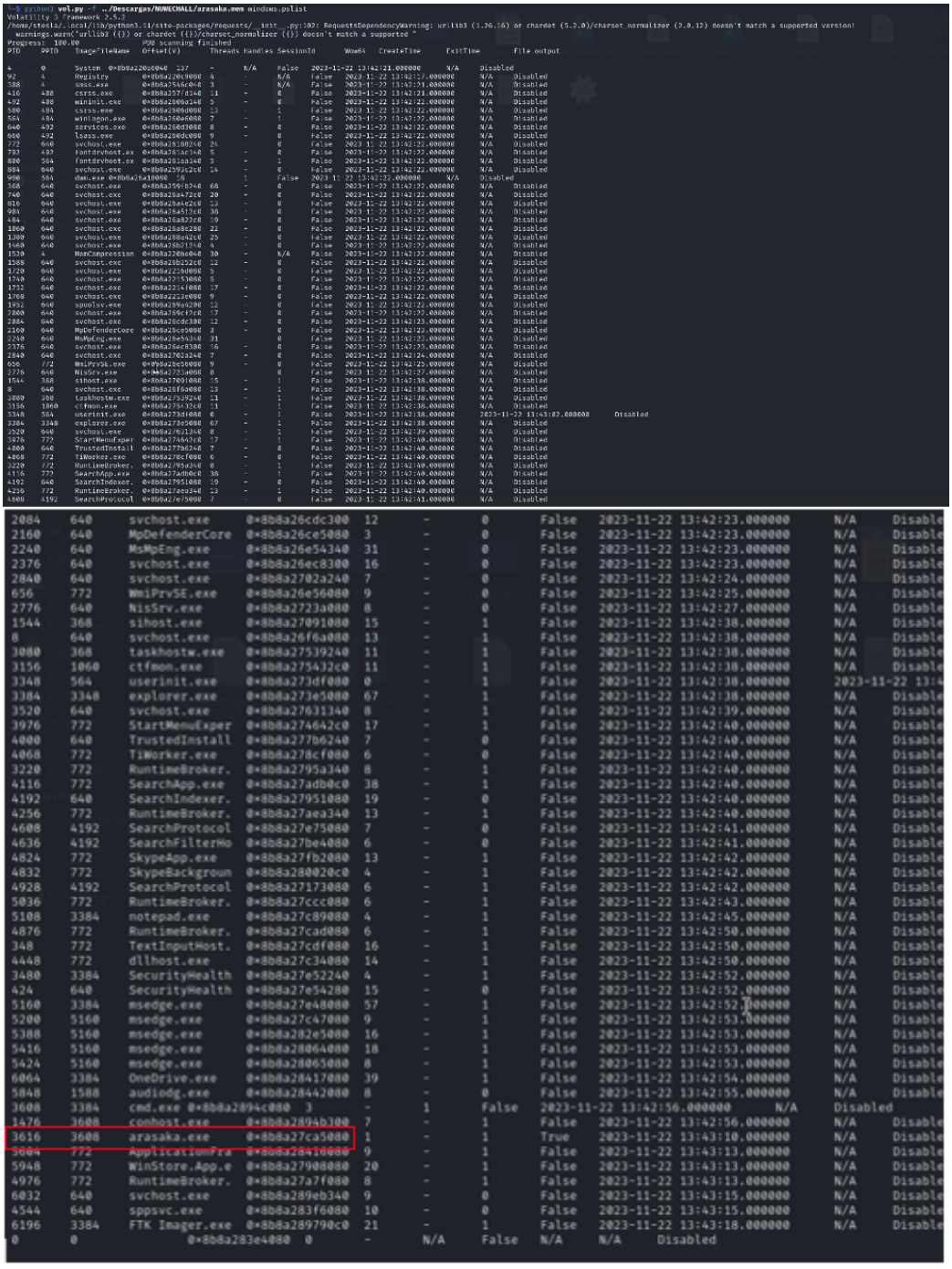


Figure 14: Obtaining the second flag using strings

Now we need to find a new way to pivot to another user or elevate privileges to root.

5 Third flag and *pivoting* to user kiwi

Using Volatility 3, we proceed to analyze the memory dump. The first thing we will do is to observe the running processes.

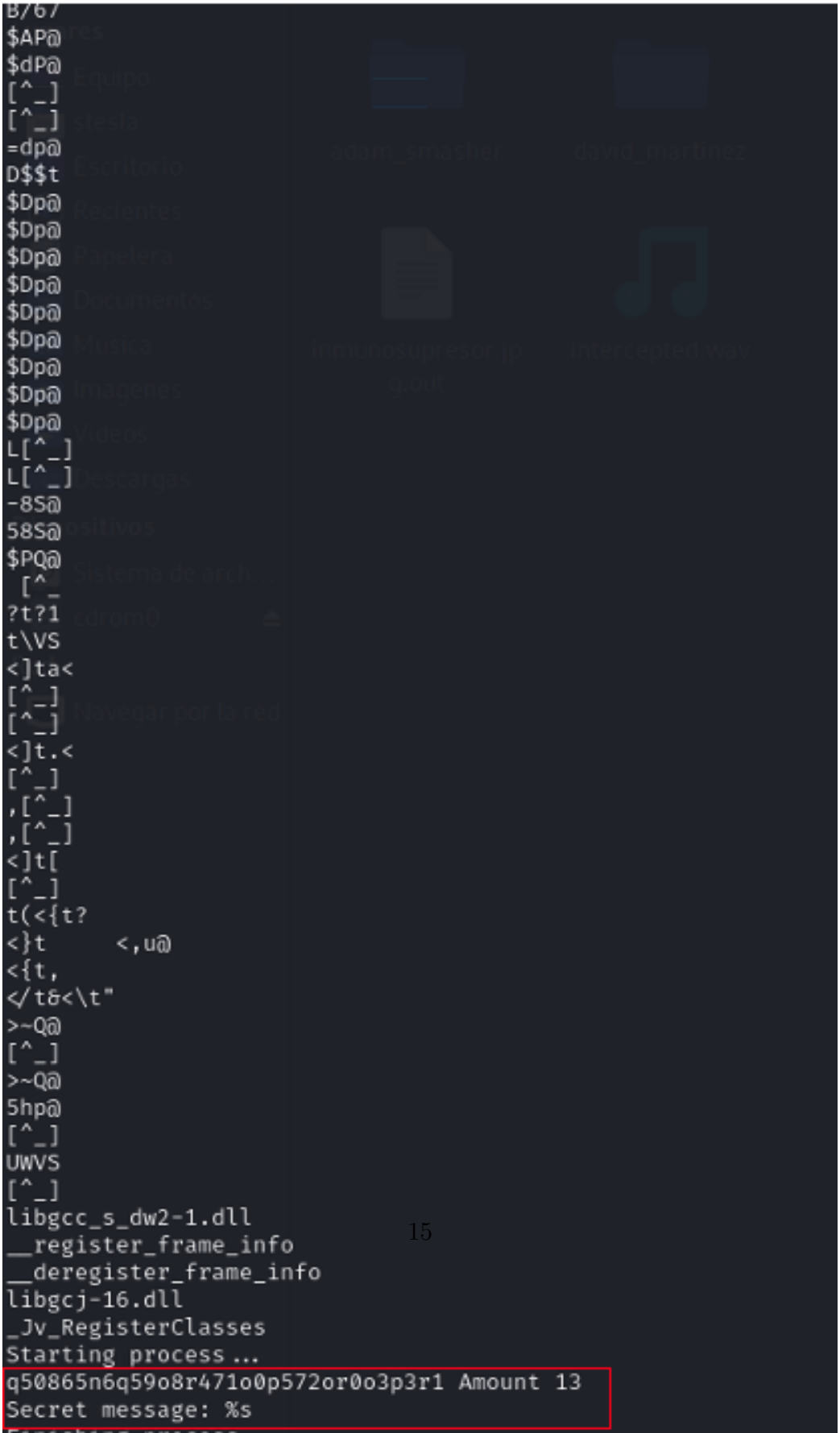


```
~$ python3 vol.py -f ../Descargas/MUWECHALL/arasaka.mem windows.pslist
Volatility 3 Framework 2.5.2
/home/stosis/.local/lib/python3.11/site-packages/requests/__init__.py:102: RequestsDependencyWarning: urllib3 (1.26.16) or chardet (5.2.0)/charset_normalizer (2.0.12) doesn't match a supported version!
  warnings.warn("urllib3 ({}) or chardet ({})/charset_normalizer ({}) doesn't match a supported "
Progress:  100.00               PDB scanning finished
PID     PPID    ImageFileName   Offset(V)       Threads Handles SessionId       Wow64   CreateTime      ExitTime        File output

4       0       System  0xb8a220c60040  137     -       N/A     False   2023-11-22 13:42:21.000000      N/A     Disabled
92      4       Registry        0xb8a220c90080  4       -       N/A     False   2023-11-22 13:42:17.000000      N/A     Disabled
388     4       smss.exe        0xb8a256a6c040  3       -       N/A     False   2023-11-22 13:42:21.000000      N/A     Disabled
416     408     csrss.exe       0xb8a2577f1240  11      -       0       False   2023-11-22 13:42:23.000000      N/A     Disabled
492     408     wininit.exe     0xb8a2006a2240  5       -       0       False   2023-11-22 13:42:22.000000      N/A     Disabled
500     484     csrss.exe       0xb8a2b0bd0080  13      -       1       False   2023-11-22 13:42:22.000000      N/A     Disabled
564     484     winlogon.exe    0xb8a256e60080  7       -       1       False   2023-11-22 13:42:23.000000      N/A     Disabled
640     492     services.exe    0xb8a2006d1080  8       -       0       False   2023-11-22 13:42:22.000000      N/A     Disabled
660     492     lsass.exe       0xb8a2b0bdc080  9       -       0       False   2023-11-22 13:42:22.000000      N/A     Disabled
772     640     svchost.exe     0xb8a2613d02c0  24      -       0       False   2023-11-22 13:42:22.000000      N/A     Disabled
792     492     fontdrvhost.ex  0xb8a261ac1c40  5       -       0       False   2023-11-22 13:42:22.000000      N/A     Disabled
800     564     fontdrvhost.ex  0xb8a2611a1a40  5       -       1       False   2023-11-22 13:42:22.000000      N/A     Disabled
884     640     svchost.exe     0xb8a2d291c2c0  14      -       0       False   2023-11-22 13:42:22.000000      N/A     Disabled
980     564     dwm.exe 0xb8a26a180080  16      1       False   2023-11-22 13:42:22.000000      N/A     Disabled
368     640     svchost.exe     0xb8a2560cb240  68      -       0       False   2023-11-22 13:42:22.000000      N/A     Disabled
740     640     svchost.exe     0xb8a26a472c80  20      -       0       False   2023-11-22 13:42:22.000000      N/A     Disabled
816     640     svchost.exe     0xb8a26a4e2c00  13      -       0       False   2023-11-22 13:42:22.000000      N/A     Disabled
984     640     svchost.exe     0xb8a26a513c80  36      -       0       False   2023-11-22 13:42:22.000000      N/A     Disabled
484     640     svchost.exe     0xb8a26a823c80  19      -       0       False   2023-11-22 13:42:22.000000      N/A     Disabled
1060    640     svchost.exe     0xb8a26a8c2300  22      -       0       False   2023-11-22 13:42:22.000000      N/A     Disabled
1300    640     svchost.exe     0xb8a26ba42c00  25      -       0       False   2023-11-22 13:42:22.000000      N/A     Disabled
1460    640     svchost.exe     0xb8a26b2172c0  4       -       0       False   2023-11-22 13:42:22.000000      N/A     Disabled
1520    4       MemCompression  0xb8a2200e6040  30      -       N/A     False   2023-11-22 13:42:22.000000      N/A     Disabled
1588    640     svchost.exe     0xb8a26b2522c0  22      -       0       False   2023-11-22 13:42:22.000000      N/A     Disabled
1720    640     svchost.exe     0xb8a2223b0080  3       -       0       False   2023-11-22 13:42:22.000000      N/A     Disabled
1740    640     svchost.exe     0xb8a22153008   5       -       0       False   2023-11-22 13:42:22.000000      N/A     Disabled
1752    640     svchost.exe     0xb8a2231410B8  17      -       0       False   2023-11-22 13:42:22.000000      N/A     Disabled
1768    640     svchost.exe     0xb8a2223e00B8  9       -       0       False   2023-11-22 13:42:22.000000      N/A     Disabled
1952    640     spoolsv.exe     0xb8a289a4200   12      -       0       False   2023-11-22 13:42:22.000000      N/A     Disabled
2000    640     svchost.exe     0xb8a2696f3c8   17      -       0       False   2023-11-22 13:42:23.000000      N/A     Disabled
2884    640     svchost.exe     0xb8a26cdc300   12      -       0       False   2023-11-22 13:42:23.000000      N/A     Disabled
2160    640     MpDefenderCore  0xb8a26ce5080   3       -       0       False   2023-11-22 13:42:23.000000      N/A     Disabled
2240    640     MsMpEng.exe     0xb8a26e54340   31      -       0       False   2023-11-22 13:42:23.000000      N/A     Disabled
2376    640     svchost.exe     0xb8a26ec8300   16      -       0       False   2023-11-22 13:42:23.000000      N/A     Disabled
2840    640     svchost.exe     0xb8a2702a240   7       -       0       False   2023-11-22 13:42:24.000000      N/A     Disabled
656     772     WmiPrvSE.exe    0xb8a26e56080   9       -       0       False   2023-11-22 13:42:25.000000      N/A     Disabled
2776    640     NisSrv.exe      0xb8a2723a080   8       -       0       False   2023-11-22 13:42:27.000000      N/A     Disabled
1544    368     sihost.exe      0xb8a27091080   15      -       1       False   2023-11-22 13:42:38.000000      N/A     Disabled
8       640     svchost.exe     0xb8a26f6a080   13      -       1       False   2023-11-22 13:42:38.000000      N/A     Disabled
3080    368     taskhostw.exe   0xb8a27539240   11      -       1       False   2023-11-22 13:42:38.000000      N/A     Disabled
3156    1060    ctfmon.exe      0xb8a275432c0   11      -       1       False   2023-11-22 13:42:38.000000      N/A     Disabled
3348    564     userinit.exe    0xb8a273df080   0       -       1       False   2023-11-22 13:42:38.000000      2023-11-22 13:4
3384    3348    explorer.exe    0xb8a273e5080   67      -       1       False   2023-11-22 13:42:38.000000      N/A     Disabled
3520    640     svchost.exe     0xb8a2763134B   8       -       1       False   2023-11-22 13:42:39.000000      N/A     Disabled
3976    772     StartMenuExper  0xb8a274642c0   17      -       1       False   2023-11-22 13:42:40.000000      N/A     Disabled
4000    640     TrustedInstall  0xb8a277b6240   7       -       0       False   2023-11-22 13:42:40.000000      N/A     Disabled
4068    772     TiWorker.exe    0xb8a278cf080   6       -       0       False   2023-11-22 13:42:40.000000      N/A     Disabled
3220    772     RuntimeBroker.  0xb8a2795a340   8       -       1       False   2023-11-22 13:42:40.000000      N/A     Disabled
4116    772     SearchApp.exe   0xb8a27adb0c0   38      -       1       False   2023-11-22 13:42:40.000000      N/A     Disabled
4192    640     SearchIndexer.  0xb8a27951080   19      -       0       False   2023-11-22 13:42:40.000000      N/A     Disabled
4256    772     RuntimeBroker.  0xb8a27aea340   13      -       1       False   2023-11-22 13:42:40.000000      N/A     Disabled
4608    4192    SearchProtocol  0xb8a27e75080   7       -       0       False   2023-11-22 13:42:41.000000      N/A     Disabled
4636    4192    SearchFilterHo  0xb8a27be4080   6       -       0       False   2023-11-22 13:42:41.000000      N/A     Disabled
4824    772     SkypeApp.exe    0xb8a27fb2080   13      -       1       False   2023-11-22 13:42:42.000000      N/A     Disabled
4832    772     SkypeBackgroun  0xb8a280020c0   4       -       1       False   2023-11-22 13:42:42.000000      N/A     Disabled
4928    4192    SearchProtocol  0xb8a27173080   6       -       1       False   2023-11-22 13:42:42.000000      N/A     Disabled
5036    772     RuntimeBroker.  0xb8a27ccc080   6       -       1       False   2023-11-22 13:42:43.000000      N/A     Disabled
5108    3384    notepad.exe     0xb8a27c89080   4       -       1       False   2023-11-22 13:42:45.000000      N/A     Disabled
4876    772     RuntimeBroker.  0xb8a27cad080   6       -       1       False   2023-11-22 13:42:50.000000      N/A     Disabled
348     772     TextInputHost.  0xb8a27cdf080   16      -       1       False   2023-11-22 13:42:50.000000      N/A     Disabled
4448    772     dllhost.exe     0xb8a27c34080   14      -       1       False   2023-11-22 13:42:50.000000      N/A     Disabled
3480    3384    SecurityHealth  0xb8a27e52240   4       -       1       False   2023-11-22 13:42:52.000000      N/A     Disabled
424     640     SecurityHealth  0xb8a27e54280   15      -       0       False   2023-11-22 13:42:52.000000      N/A     Disabled
5160    3384    msedge.exe      0xb8a27e48080   57      -       1       False   2023-11-22 13:42:52.000000      N/A     Disabled
5200    5160    msedge.exe      0xb8a27c47080   9       -       1       False   2023-11-22 13:42:53.000000      N/A     Disabled
5388    5160    msedge.exe      0xb8a282e5080   16      -       1       False   2023-11-22 13:42:53.000000      N/A     Disabled
5416    5160    msedge.exe      0xb8a28064080   18      -       1       False   2023-11-22 13:42:53.000000      N/A     Disabled
5424    5160    msedge.exe      0xb8a28065080   8       -       1       False   2023-11-22 13:42:53.000000      N/A     Disabled
6064    3384    OneDrive.exe    0xb8a28417080   39      -       1       False   2023-11-22 13:42:54.000000      N/A     Disabled
5048    1508    audiodg.exe     0xb8a28442080   8       -       0       False   2023-11-22 13:42:55.000000      N/A     Disabled
3608    3384    cmd.exe 0xb8a2894c080   3       -       1       False   2023-11-22 13:42:56.000000      N/A     Disabled
1476    3608    conhost.exe     0xb8a2894b300   7       -       1       False   2023-11-22 13:42:56.000000      N/A     Disable
3616    3608    arasaka.exe     0xb8a27ca5080   1       -       1       True    2023-11-22 13:43:10.000000      N/A     Disable
3604    772     ApplicationFra  0xb8a28410080   9       -       1       False   2023-11-22 13:43:13.000000      N/A     Disable
5948    772     WinStore.App.e  0xb8a27908080   20      -       1       False   2023-11-22 13:43:13.000000      N/A     Disable
4976    772     RuntimeBroker.  0xb8a27a7f080   8       -       1       False   2023-11-22 13:43:13.000000      N/A     Disable
6032    640     svchost.exe     0xb8a289eb340   9       -       0       False   2023-11-22 13:43:15.000000      N/A     Disable
4544    640     sppsvc.exe      0xb8a283f6080   10      -       0       False   2023-11-22 13:43:15.000000      N/A     Disable
6196    3384    FTK Imager.exe  0xb8a289790c0   21      -       1       False   2023-11-22 13:43:18.000000      N/A     Disable
0       0               0xb8a283e4080   0       -       N/A     False   N/A     N/A     Disabled
```

Figure 15: In-memory process listing with Volatility 3

We see that there is an interesting executable: *arasaka.exe*, whose name makes us suspect that it also has to do with the challenge. We dump this file using the command *pslist –dump –pid 3616* of Volatility 3 and, again, we apply strings on the extracted binary. If we look closely, we notice that there is an odd string next to the "Amount 13" phrase, which makes us suspect that a 13-position rotation has been applied to the characters in the string.


B767
$AP@
$dP@
[^_]
[^_]
=dp@
D$$t
$Dp@
$Dp@
$Dp@
$Dp@
$Dp@
$Dp@
$Dp@
$Dp@
$Dp@
$Dp@
L[^_]
L[^_]
-8S@
58S@
$PQ@
[^_
?t?1
t\VS
<]ta<
[^_]
[^_]
<]t.<
[^_]
,[^_]
,[^_]
<]t[
[^_]
t(<{t?
<}t <,u@
<{t,
</t&<\t"
>~Q@
[^_]
>~Q@
5hp@
[^_]
UWVS
[^_]
libgcc_s_dw2-1.dll
__register_frame_info
__deregister_frame_info
libgcj-16.dll
_Jv_RegisterClasses
Starting process ...
q50865n6q59o8r471o0p572or0o3p3r1 Amount 13
Secret message: %s

Using the Cyberchef tool, we apply this rotation and get what appears to be a hash:

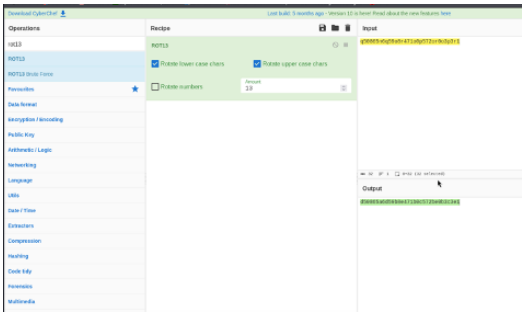


Figure 17: Hash obtained after rotation.

Intuition tells us that it looks like the goal will be to crack this hash, so we take it to the Crackstation service and, sure enough, we get the plaintext associated with that hash:

Figure 18: Result of the plaintext associated with the hash.

Using this string as the password for the user kiwi, we manage to impersonate this user. Again, we make a copy of the user's folder on our attacking machine to ensure its availability. We see that, this time, there are several interesting folders and files in the user's home. We will focus first on the hidden folder *.auth_bin*, which contains an executable binary called *kiwilidator*. The first thing we will do is to analyze it with the Ghidra software. The code we are most interested in is the *main* function.

```
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();
  }
```

Figure 19: Main function of the binary kiwilidator.

As we can see, the following actions are performed in this function:

1. Read user input.

2. Apply the OpenSSL sha256 hash function to it.

3. Compare the resulting hash with a hash stored in memory.

4. In case of a match, the generate flag function is called which will use the hash of the user input to generate it.

Therefore, to overcome this comparison and get the correct flag, we must get our input to generate the correct sha256 hash, which would involve cracking the hash provided, or patching the program to make the hash associated with our input have the expected value at the time of the comparison. We will opt for this second option as it is faster and does not depend on the computational capacity to perpetrate a brute force attack. We will perform the task with GDB. We set a breakpoint just before the call to memcomp and place in the RDI register the expected value for the hash of our input:

Figure 20: RDI patching to match hash values.

And in this way we get that, when the function *generate_flag* is called, it is generated correctly.

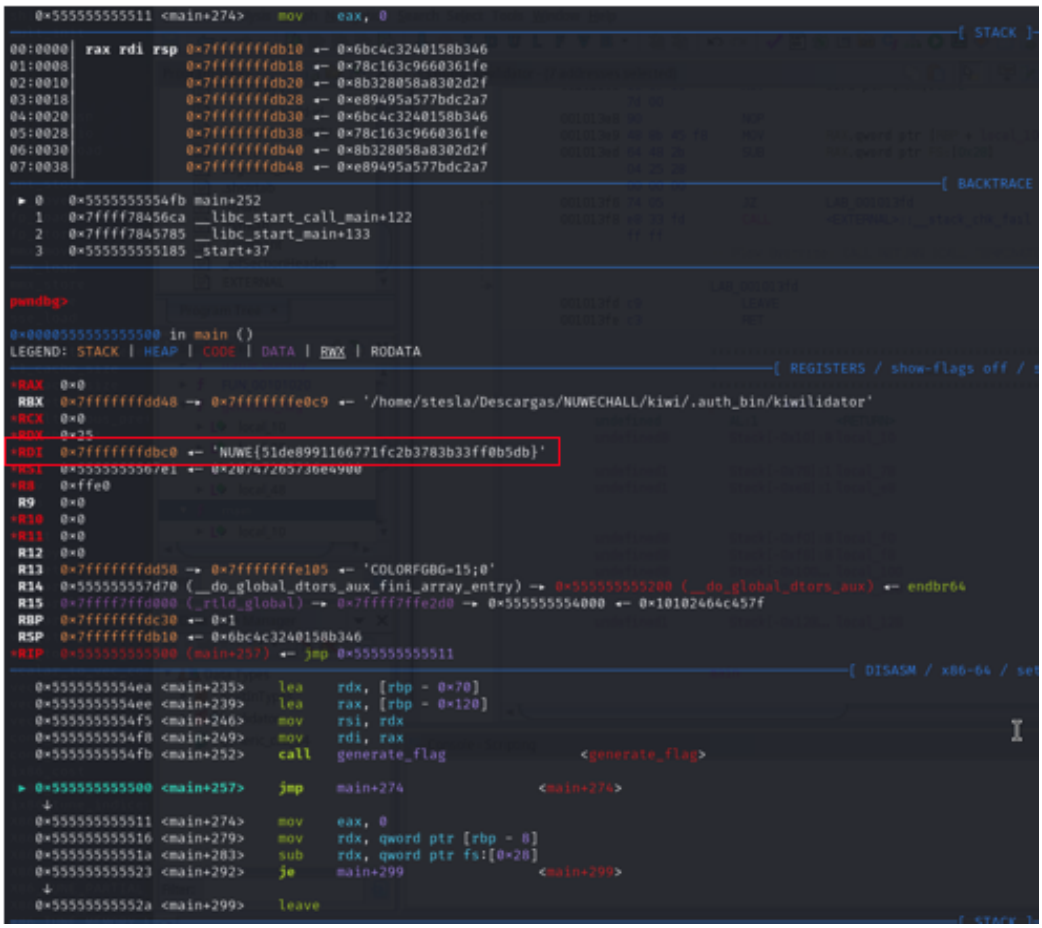


Figure 21: Obtaining the kiwi user's flag.

6 Fourth flag and *pivoting* to user Adam Smasher

Once we have obtained the flag, we check the kiwi user's folders again. In the csv file, we see what appears to be a list of system user passwords, noting that the last password for *adam_smasher* is unknown. If we launch the *exiftool* tool on all the images found under the *hacked_cams* directory, we observe something curious in one of them: it has been modified with

Photoshop. Since all the images appear to be generated with an AI, this detail is striking, as it seems to indicate that content has been added on purpose.

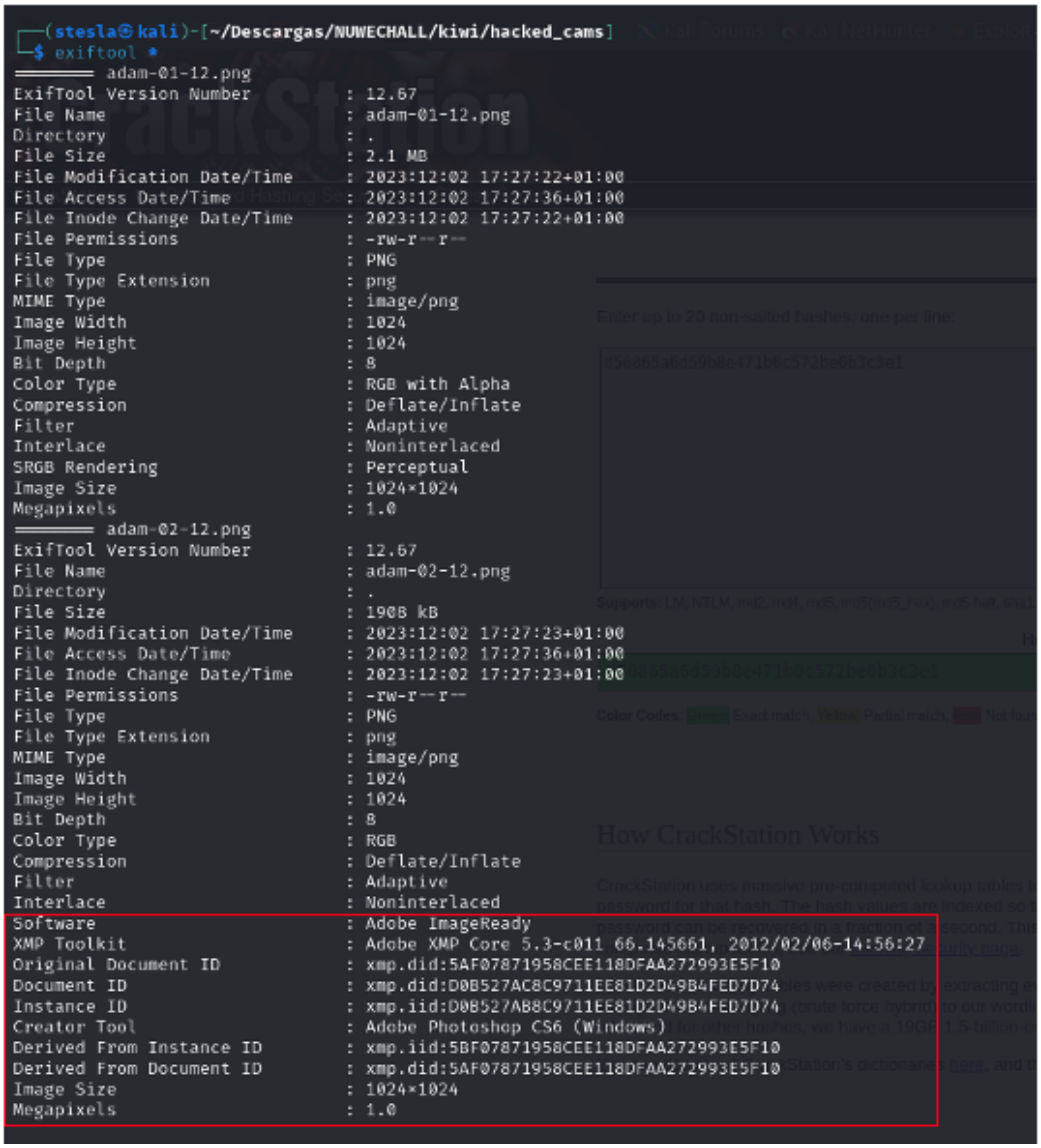


Figure 22: Execution of exiftool on the folder *hacked_cams*

If we open this image, we notice that, on the paper you can see, the user's

password *adam_smasher* seems to be written (the string starting with smash seems to be a clear indication of this).

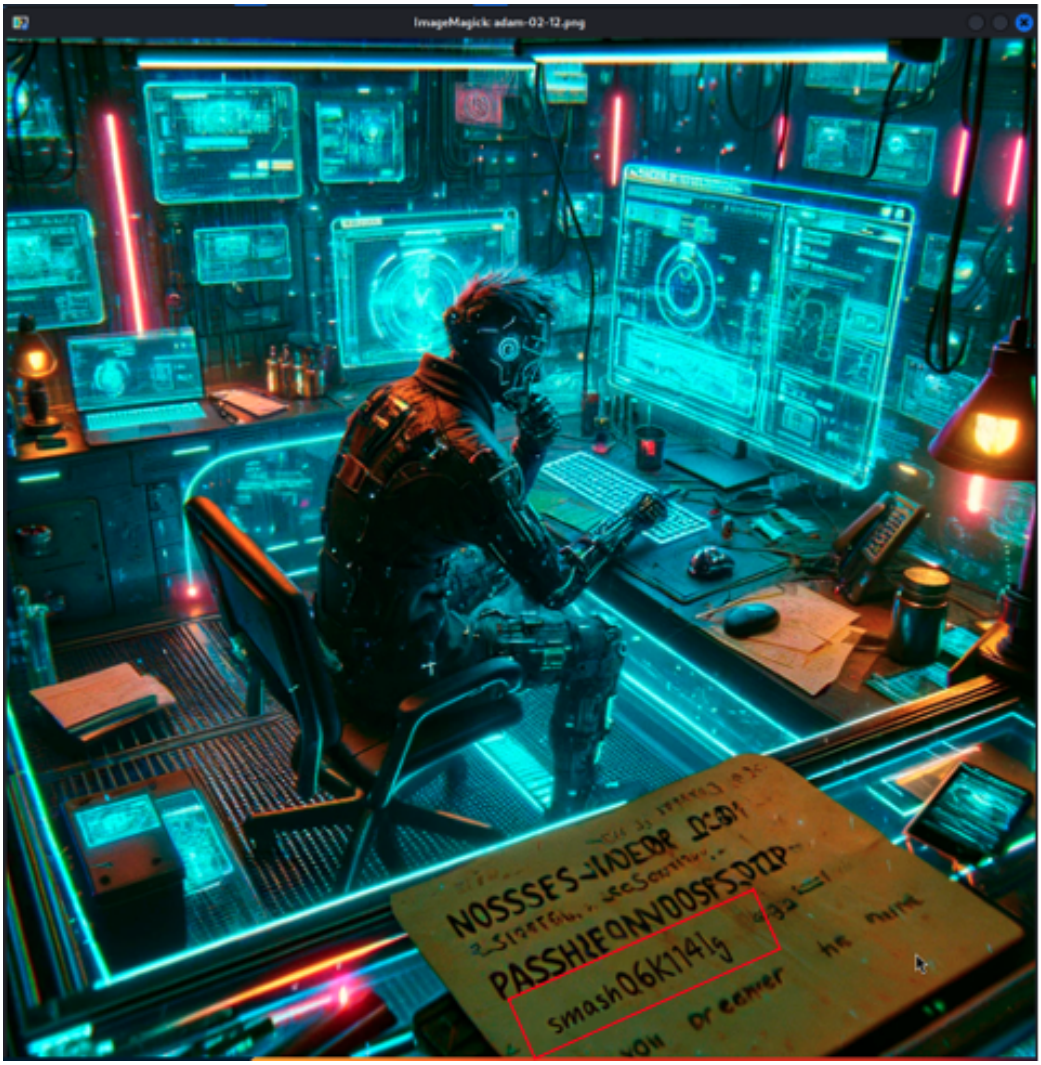


Figure 23: Obtaining adam's password.

Once we get adam's password, we impersonate him and get the flag associated with his user. Again, we copy the contents of his home directory to our local machine.

7 Fifth flag and elevation of root privileges

After gaining access to the adam user, we see that there is not much more to analyze in the home directory. After a while, we decide to use the tool pspy64, which allows us to monitor the processes running on Linux machines without having root permissions. In Figure 24 we observe something interesting: the root user is going to execute a file included in the directory *.target* of our current user. The execution takes place between *sleeps* and proceeds to run the file in execution and create it again continuously, so, if we want the contents of the file to be executed as root we must be fast or generate a race condition.

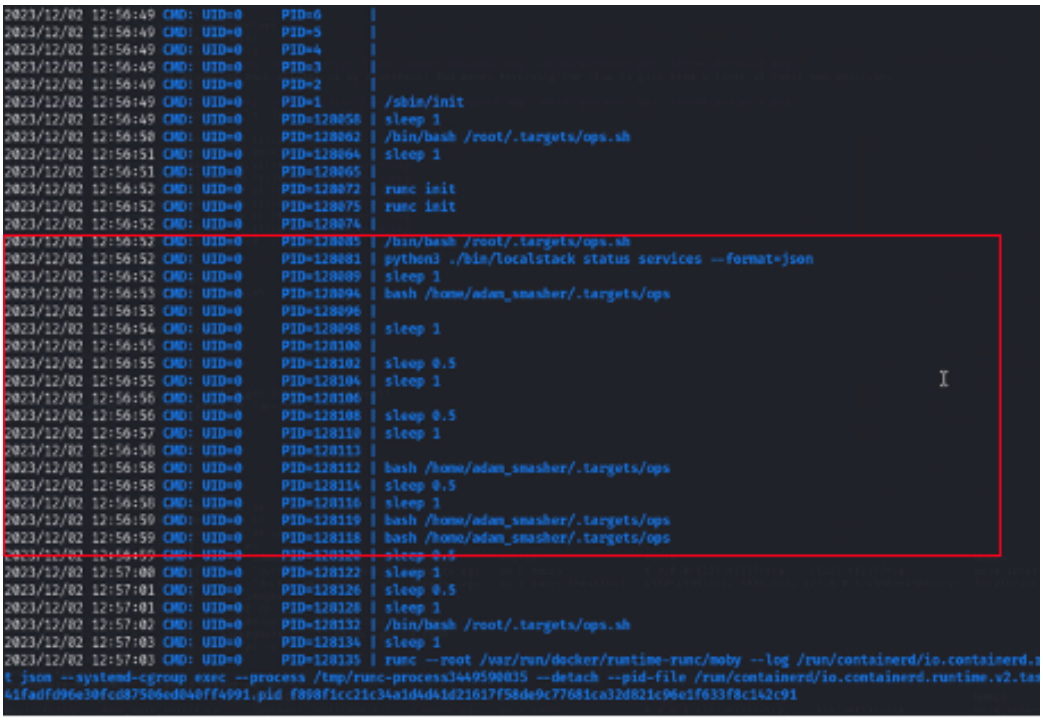


Figure 24: Process monitoring with pspy64

To generate such a race condition, we run an infinite loop in bash that takes care of writing a reverse shell in the *ops* file, and we place a netcat process listening on our local machine using ngrok to tunnel the connection from the internet to our machine. Eventually, we will receive the remote connection as root user:

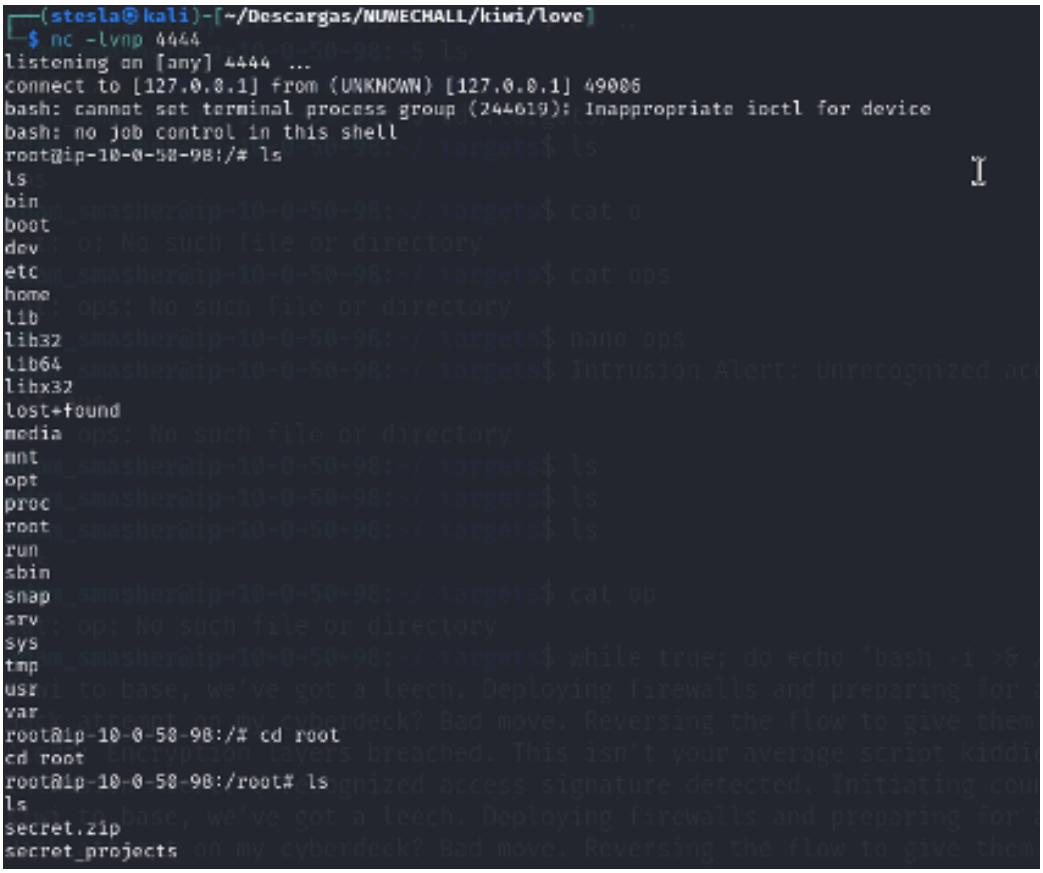


Figure 25: Full privilege escalation.

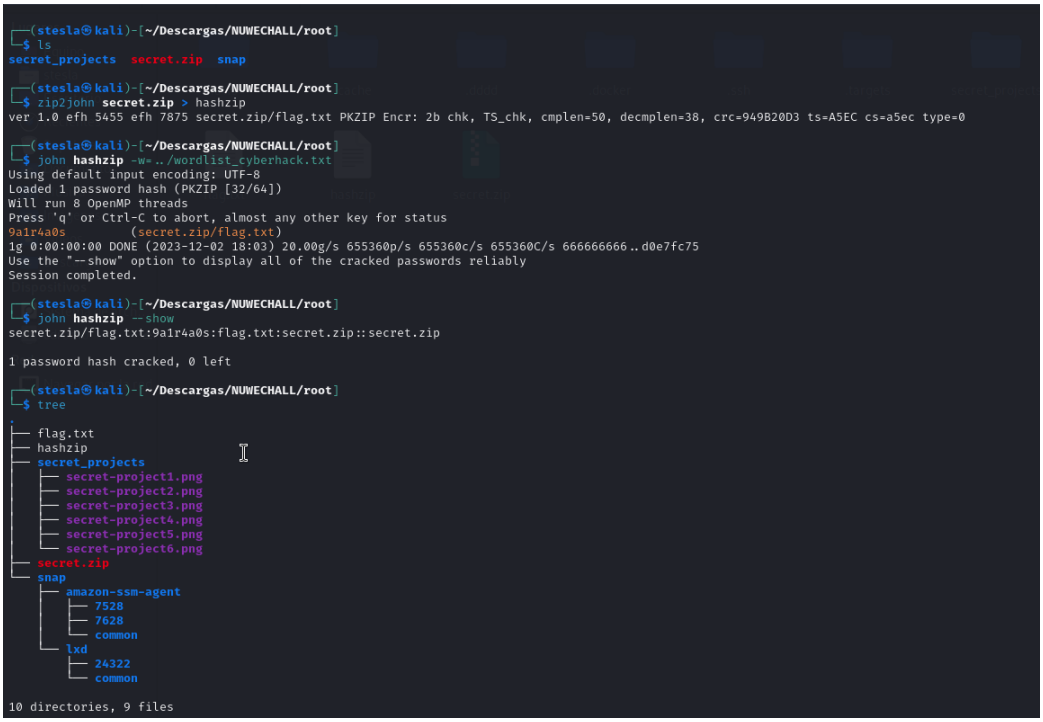


Figure 26: Result of cracking the zip file.

As we can see, we find in the root directory a zip file. When we open it, we see that it is a compressed file containing the password-protected flag. As specified in Section 1, Nuwe provides us with a wordlist, so the procedure is clear: we need to crack the zip file to obtain the password. To perform this task, we will make use of the zip2john utility and the John the Ripper tool.

With this process, we obtain the root user flag.

8 Sixth and last flag of user David Martínez

Now that we have access to the root user, we can query any file on the system. The only user we have left to analyze is *david_martinez*, so we access his home folder and see two files: *farewell* and *crypt*. If we open the *farewell* file, we see that the content seems to have been encrypted with some kind of shift algorithm. If we analyze the code of the *crypt* file we will notice that a 19-position rotation has been applied to the characters. Performing the

same rotation in Cyberchef, we obtain the last flag of the challenge.

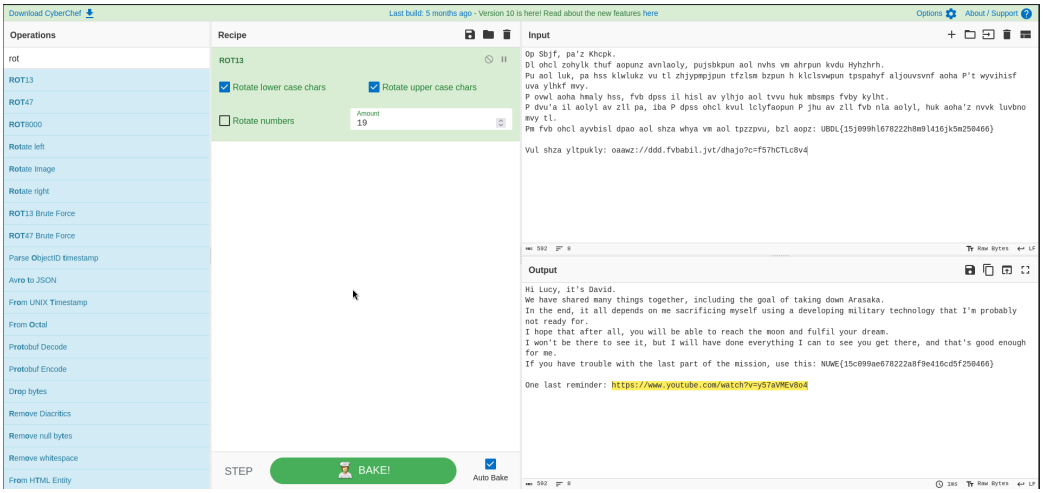


Figure 27: Flag of David Martinez and end of the challenge

9 Conclusions

In this interesting challenge proposed by Nuwe we have exploited different vulnerabilities and applied different system penetration techniques, as well as reverse engineering and forensic analysis to advance through the system, pivoting between users and, finally, gaining full control of the machine by obtaining root privileges. It has been quite an interesting challenge, although I must admit that it required quite a few hours as the only member of my team. Even so, the experience has been quite satisfactory. I would like to thank the Nuwe team for the initiative.

Figure 28: GGWP!