Here’s a detailed breakdown of **how to solve the challenge** along with the **main tools and techniques** to efficiently identify the malicious file:

**1. Understand the Challenge**

• **Scenario**: Detect a single malicious file among hundreds using an API provided by a web-based antivirus.

• **Tools**:

• Bash scripting

• curl for API interactions

• Basic Linux CLI (e.g., loops, file handling)

• ClamAV API

**2. Tools and Techniques**

**2.1 API Communication Using curl**

The API allows:

• Authentication to generate a token.

• Submitting files for scanning.

**Key** curl **flags**:

• -k: Ignore SSL certificate validation (useful for self-signed certs).

• -H: Add headers like Authorization.

• -F: Upload files in multipart/form-data format.

• -X: Specify HTTP method (e.g., POST).

**Authentication Example**:

curl -k https://clamav-ui.com/api/v1/auth

**File Scan Example**:

curl -X POST -k "https://clamav-ui.com/api/v1/scan" \

-H 'Authorization: YOUR\_TOKEN' \

-H "Content-Type: multipart/form-data" \

-F "file=@./suspicious-files/file0.exe"

**2.2 Scripting with Bash**

To automate file scanning for hundreds of files, Bash scripting is ideal due to its lightweight nature.

**Steps to Script Automation:**

1. **Create a loop to iterate through all files**:

• Use a while or for loop.

• Change the filename dynamically for each iteration.

2. **Generate the Token**:

• Use a variable to store the token to simplify the script.

3. **Build API Requests**:

• Use curl with the correct headers and file paths.

4. **Save Results**:

• Append scan results to a log file for further filtering.

**Example Script**:

#!/bin/bash

i=0 *# Counter for files*

file=272 *# Total number of files*

token="YOUR\_API\_TOKEN" *# Replace with actual token*

log="script.log"

*# Clear or create log file*

echo "" > $log

*# Loop through all files and send API requests*

while [ $i -lt $file ]; do

echo "Scanning file $i" >> $log

curl -X POST -k "https://clamav-ui.com/api/v1/scan" \

-H "Authorization: $token" \

-H "Content-Type: multipart/form-data" \

-F "file=@./suspicious-files/file$i.exe" >> $log

i=$((i + 1))

done

5. **Run the Script**:

• Grant execute permissions using chmod:

chmod 711 script.sh

• Execute the script:

./script.sh

**2.3 Filtering Results**

Once the log file is generated, you need to isolate files that show the **“infectedFiles”** key in the response.

**Command to Filter**:

cat script.log | grep "infectedFiles"

This will return lines containing the keyword infectedFiles, indicating the malicious file.

**2.4 Submitting the Flag**

• Once you identify the malicious file:

• Note its **MD5 hash**.

• Submit it as the flag.

**Manual Verification**:

• Upload the file manually to the ClamAV web interface for confirmation.

**3. Tips to Solve the Challenge**

• **Efficiency**: Automate the process using loops and variables to avoid manual file uploads.

• **Debugging**: Add echo statements to track progress during script execution.

• **Error Handling**: Ensure the script handles API errors (e.g., invalid tokens, file upload failures).

• **Testing**: Start with a smaller subset of files (e.g., first 5 files) to test the script before running it on all 272 files.

**4. Tools to Use**

• **Bash Shell**: Native Linux scripting.

• **curl**: For API communication.

• **grep**: To filter and analyze log files.

• **Visual Studio Code**: A convenient IDE for scripting.

By following this systematic approach, you can solve the challenge efficiently, identify the malicious file, and submit the MD5 hash as the flag. Let me know if you need further clarification or assistance! 🚀