Threat Intelligence Task

Scenario Overview

You are part of a cybersecurity threat intelligence team tasked with identifying a potential cyber threat hidden among multiple hash values. A recent report suggests that a well-known malware strain has resurfaced, but intelligence sources have mixed it with several random hashes. Your mission is to analyze the given hash dataset, identify any malicious sample, investigate its history, and create a YARA rule to detect it in future attacks.

Task Breakdown

1. Identifying the Malicious Hash Using VirusTotal

You are provided with a list of 50 SHA-256 hash values. You must:

- Use Python and the **requests** library to interact with the VirusTotal API.
- Analyze each hash to determine whether it is associated with a known malware threat.
- Extract and document relevant details from VirusTotal's response.

2. Threat Intelligence Research

Once you identify a malicious hash, conduct thorough research and document the following:

- **History of the malware:** When did it emerge?
- **Notable Attacks:** What organizations or industries were targeted?
- **Threat Actor:** Who is behind the malware? Any known affiliations?
- **Motivation & Tactics:** Why is this malware used, and what techniques does it employ?
- **Attack Lifecycle:** How does it infect systems and execute its objectives?
- **Current Status:** Is the malware still active? Have security measures mitigated it?
- **Detection & Prevention Strategies:** What security controls can defend against it?

Prepare a detailed report summarizing your findings.

3. YARA Rule Creation

Using the identified hash and intelligence gathered, create a **YARA rule** to detect samples associated with the malware and test the rule. Your rule will include:

- **SHA-256 hash** (from the detected malicious hash value).

- **Suspicious Strings** found in the malware, such as unique file names, hardcoded URLs, or registry entries.
- **File Characteristics** like common extensions, headers, or magic bytes.
- **Behavioral Indicators**, e.g., registry modifications, process execution patterns, or mutex creation.

4. Reporting: Prepare a comprehensive threat intelligence report documenting your analysis and findings. Present your findings to executive leadership and relevant stakeholders, emphasizing the importance of proactive cybersecurity measures in mitigating such attacks based on your analysis.

🔹 Step 1: Install Required Packages

Since **requests** is available in Kali's package repositories, install it using:

```
sudo apt update && sudo apt install python3-venv python3-requests -y
```

🔹 Step 2: Get a VirusTotal API Key

You'll need an API key to check hashes against VirusTotal.

1. Sign up at VirusTotal.
2. Log in and go to your **profile settings**.
3. Copy your **API key** (you'll use it in the script).

🔹 Step 3: Create a Python Script to Check Hashes

1️⃣Open a terminal and create the Python script:

nano check_hashes.py

2️⃣Copy and paste the script at the later part of this document:

(Replace **"YOUR_VIRUSTOTAL_API_KEY"** *with your actual API key.)*

🔹 Step 4: Run the Script

1️⃣Run the script to check hashes against VirusTotal:

python3 check_hashes.py

🔹 Python script to be added to the **check_hashes.py** file:

```
import requests
import time

API_KEY = "YOUR_VIRUSTOTAL_API_KEY"  # Replace with your actual API key
hashes = [

    "Aad36e1a3e40e4e5a62b5b12bcd227b8", # Replace with the actual 50 hashes
    "f2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2",
    "5d41402abc4b2a76b9719d911017c592",
    "6a7b8c9d0e1f2g3h4i5j6k7l8m9n0o1p"
]

for index, file_hash in enumerate(hashes):
    url = f"https://www.virustotal.com/api/v3/files/{file_hash}"
    headers = {"x-apikey": API_KEY}

    response = requests.get(url, headers=headers)
    data = response.json()

    if "data" in data and "attributes" in data["data"]:
        stats = data["data"]["attributes"]["last_analysis_stats"]
        vendors = data["data"]["attributes"].get("last_analysis_results", {})

        if stats["malicious"] > 0:
            detected_by = []
            for vendor, result in vendors.items():
                if result["category"] == "malicious":
                    detected_by.append(f"{vendor}: {result['result']}")

            print(f"⚠️ {file_hash} is flagged as malicious by:")
            for detection in detected_by:
                print(f"   - {detection}")

        else:
            print(f"✅ {file_hash} appears clean.")

    if (index + 1) % 4 == 0:
        print("⏳ Waiting for 60 seconds to stay within API rate limit...")
        time.sleep(60)
```

🔹 Hash Values to be added to the **check_hashes.py** file

```
"9f2b4c6d7e8a3f1c5b9d2e0f4a7c3b6e8d1f2a5c9b0d7e3f6a2c4b8e9d1f51c6",
"3d8c9a2b6e7d1f0a5c3b9e8d7f6a2c9b4e3d1f5a7c6e0b9d2f4a3c1e8b7d62f5",
"156335b95ba216456f1ac0894b7b9d6ad95404ac7df447941f21646ca0090673",
"6d7e8a3f1c5b9d2e0f4a7c3b6e8d1f2a5c9b0d7e3f6a2c4b8e9d1f5a9f2b47e8",
"0f4a7c3b6e8d1f2a5c9b0d7e3f6a2c4b8e9d1f5a9f2b4c6d7e8a3f1c5b9d26a4",
"2f6a2c4b8e9d1f5a9f2b4c6d7e8a3f1c5b9d2e0f4a7c3b6e8d1f2a5c9b06f2e4",
"156335b95ba216456f1ac0894b7b9d6ad95404ac7df447946f21646ca0090673",
"7e3f6a2c4b8e9d1f5a9f2b4c6d7e8a3f1c5b9d2e0f4a7c3b6e8d1f2a5c9b05d6",
"5b9d2e0f4a7c3b6e8d1f2a5c9b0d7e3f6a2c4b8e9d1f5a9f2b4c6d7e8a3f13d7",
"6a2c4b8e9d1f5a9f2b4c6d7e8a3f1c5b9d2e0f4a7c3b6e8d1f2a5c9b0d7e35f1",
"156335b95ba216456f1ac0894b7b9d6ad95404ac7df447948f21646ca0090673",
"8d7e8a3f1c5b9d2e0f4a7c3b6e8d1f2a5c9b0d7e3f6a2c4b8e9d1f5a9f2b40e3",
"1f5a9f2b4c6d7e8a3f1c5b9d2e0f4a7c3b6e8d1f2a5c9b0d7e3f6a2c4b8e97c5",
"156335b95ba216456f1ac0894b7b9d6ad95404ac7df447943f21646ca0090673",
"5c9b0d7e3f6a2c4b8e9d1f5a9f2b4c6d7e8a3f1c5b9d2e0f4a7c3b6e8d1f28a9",
"156335b95ba216456f1ac0894b7b9d6ad95404ac7df447945f21646ca0090673",
"9e9d1f5a9f2b4c6d7e8a3f1c5b9d2e0f4a7c3b6e8d1f2a5c9b0d7e3f6a2c42d8",
"0a9f2b4c6d7e8a3f1c5b9d2e0f4a7c3b6e8d1f2a5c9b0d7e3f6a2c4b8e9d12f6",
"156335b95ba216456f1ac0894b7b9d6ad95404ac7df447940f21646ca0090673",
"6c4b8e9d1f5a9f2b4c6d7e8a3f1c5b9d2e0f4a7c3b6e8d1f2a5c9b0d7e3f51b9",
"156335b95ba216456f1ac0894b7b9d6ad95404ac7df447947f21646ca0090673",
"156335b95ba216456f1ac0894b7b9d6ad95404ac7df447944f21646ca0090673",
"7a3f1c5b9d2e0f4a7c3b6e8d1f2a5c9b0d7e3f6a2c4b8e9d1f5a9f2b4c6d72e5",
"1d5a9f2b4c6d7e8a3f1c5b9d2e0f4a7c3b6e8d1f2a5c9b0d7e3f6a2c4b8e94a7",
"5e8d1f2a5c9b0d7e3f6a2c4b8e9d1f5a9f2b4c6d7e8a3f1c5b9d2e0f4a7c34c8",
"1f5a9f2b4c6d7e8a3f1c5b9d2e0f4a7c3b6e8d1f2a5c9b0d7e3f6a2c4b8e93a7",
"156335b95ba216456f1ac0894b7b9d6ad95404ac7df447942f21646ca0090673",
"9b2e0f4a7c3b6e8d1f2a5c9b0d7e3f6a2c4b8e9d1f5a9f2b4c6d7e8a3f1c58d3",
"7e3f6a2c4b8e9d1f5a9f2b4c6d7e8a3f1c5b9d2e0f4a7c3b6e8d1f2a5c9b05d6",
"6c6d7e8a3f1c5b9d2e0f4a7c3b6e8d1f2a5c9b0d7e3f6a2c4b8e9d1f5a9f20f4",
"5b8d1f2a5c9b0d7e3f6a2c4b8e9d1f5a9f2b4c6d7e8a3f1c5b9d2e0f4a7c13e9"
```

How To Install Yara and Create Yara Rules