USABILITY TEST REPORT

1. Introduction

System Definition: The developed system is a Python-based **"MIRS" (Malware Incident Response System)** automation tool. It utilizes the `Watchdog` library to monitor a specific directory (`izlenen`) in real-time. The system calculates the SHA-256 hash of new files, queries the **VirusTotal API** for threat intelligence, and automatically moves malicious files to a `quarantine` directory. All actions are recorded in a local log file (`olay_gunlugu.txt`) with timestamps.

2. Users

Demographics of the 3 users recruited for the test:

- **User 1 (Student):** 21 years old, Male. Computer Engineering. (Understands the source code and logic).
- **User 2 (Student):** 23 years old, Male. Aerospace Engineer
- **User 3 (Student):** 23 years old, Male. Non-technical background.

3. Tasks (Use Cases)

The tasks included in the usability test were:

1. **Activate Monitoring:** Run the script and successfully initiate the `izlenen` (watch) directory monitoring.
2. **Malware Simulation:** Drop a file named `test_virusu.exe` (or a dummy file) into the folder and observe the system's "Hollywood Effect" (simulated analysis steps) and final alarm.
3. **API Stress Test:** Copy multiple files rapidly to test how the system handles VirusTotal API rate limits (HTTP 429 errors).
4. **Quarantine Verification:** Check if the system correctly renames files to avoid overwriting if a file with the same name already exists in the `quarantine` folder.

4. Method

- **Test Procedure:** A "Black Box" testing approach was used. Users were asked to interact with the file system while the script ran in the terminal. The "Think Aloud" protocol was applied, requiring users to verbalize their thoughts during the process.
- **Environment:** Windows 10/11 Environment, VS Code Terminal.
- **Recording Method:** Screen recording and manual note-taking of terminal outputs.

5. Results

Task Completion Status & Time:

- **Completion Rate:** 100% of users successfully initiated the scan.
- **Time Distribution:** Users spent an average of **8-10 seconds** per file analysis. This duration was intentional due to the `time.sleep` functions (the "Hollywood Effect") added to the code, which users reported made the analysis feel "deeper" and more professional.

Satisfaction Questionnaire Result:

- **User Feedback:** "The step-by-step terminal logs (e.g., 'Calculating Hash...', 'Querying API...') provided a sense of security."
- **Score:** 4.2 / 5 (Average Satisfaction).

6. Conclusion: Detected Problems & Solutions

This section reports the problems detected during the test and the code-level solutions applied.

Problem 1: Quarantine File Conflict (Overwriting Issue)

- **Description:** During testing, when a second malicious file with the same name (e.g., `malware.exe`) was detected, the system either failed to move it or overwrote the previous evidence in the `quarantine` folder.
- **Source:** The `shutil.move` function does not handle duplicate filenames by default.

Solution 1:

- **Action:** Logic was added to the `karantinaya_al` function to check if the file exists. If it does, the current timestamp is prepended to the filename.

Problem 2: System Hang on Network Timeout

- Description: When a user disconnected from the internet, the program froze indefinitely while waiting for a response from VirusTotal.
- Source: The `requests.get` function in `virustotal_sorgula` lacked a `timeout` parameter.

Solution 2:

- Action: A 10-second timeout was added to the API request, and a `try-except` block was implemented to catch `requests.exceptions.Timeout`.

Problem 3: Undefined Function Error (NameError)

- Description: The script crashed with `NameError: name 'log_to_gui' is not defined` during the "Hollywood Effect" sequence.
- Source: The code attempted to call a function named `log_to_gui` for visual effects, but this function was not defined in the provided scope.
- Solution: The missing function was replaced with standard `print` statements or mapped to the existing `log_yaz` function to ensure stability.

Problem 4: API Rate Limiting (HTTP 429)

- Description: When testing with multiple files, the VirusTotal API returned a "429 Too Many Requests" error, which the system initially treated as a generic failure.
- Source: Lack of specific handling for the HTTP 429 status code.

Solution 4:

- Action: An `elif` block was added to handle status code 429 explicitly, warning the user about the quota limit.