**Use Case Documentation**

**Antivirus: AI Based Threat Detection**

Version 1.0

Prepared by [**Usman, Yameen and Daniyal**]

[**Iqra University – Airport Campus**]

[**1st May 2025**]

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **UC01** | | |
| Use Case Name: | **Start Scan** | | |
| Created By: | Usman | Last Updated By: | Usman |
| Date Created: | 1st May, 2025 | Date Last Updated: | 1st May 2025 |

**Use Case 1: UC01 – Start Scan**

**Actor:** User

**Description:** The user initiates a malware scan by selecting a directory.

**Preconditions:**

1. Antivirus application is running.
2. User has permission to access the selected directory.
3. The system environment is ready (no missing dependencies or crashed modules).

**Postconditions:**

1. Scan process begins for the selected directory.
2. The directory path is passed to the file scanning module (UC02).

**Priority:** High

**Frequency of Use:** Multiple times per day.

**Normal Course of Events:**

1. User opens the antivirus application.
2. User selects “Start Scan” from the interface or terminal.
3. File explorer or terminal prompts for a directory path.
4. User selects or types a valid directory path.
5. System validates the path.
6. System initiates UC02 (Scan Files) for that directory.

**Alternative Courses:**

* **UC01.AC.1:** If the selected directory is invalid, system notifies the user and prompts for another selection.
* **UC01.AC.2:** User cancels the scan request before selection, process aborts cleanly.

**Exceptions:**

* **UC01. EX.1:** System crashes, an error log is generated and stored locally.
* **UC01. EX.2:** File explorer fails to open (in GUI mode), fallback to manual input.

**Includes:** UC02 (Scan Files).

**Special Requirements:**

1. Real-time feedback must be provided before continuing.
2. Path selection should support drag-drop (GUI) or auto-complete (CLI).
3. Directory path must handle spaces and special characters.

**Assumptions:**

1. User understands basic file navigation.
2. User has permissions for the selected folder.

**Notes and Issues:**

* On systems with high User Account Control (UAC), access to protected directories may be denied, system must handle this gracefully.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **UC02** | | |
| Use Case Name: | **Scan Files** | | |
| Created By: | Usman | Last Updated By: | Usman |
| Date Created: | 1st May, 2025 | Date Last Updated: | 1st May, 2025 |

**Use Case 2: UC02 – Scan Files**

**Actor:** System

**Description:** System recursively reads and lists all files in the selected directory.

**Preconditions:**

1. Directory selected in UC01.

**Postconditions:**

1. Complete file list generated for scanning.

**Priority:** High

**Frequency of Use:** Matches UC01.

**Normal Course of Events:**

1. System traverses the directory.
2. System identifies all files.
3. Each file path is logged internally.
4. Triggers UC03 (Extract File Features) and UC04 (Pattern Matching).

**Alternative Courses:**

* **UC02.AC.1:** If symbolic link is encountered, it is skipped to prevent loops.

**Exceptions:**

* **UC02. EX.1:** Permission denied for a file; skip and log error.
* **UC02. EX.2:** Unreadable or broken file path; log and continue.

**Includes:**

1. UC03 (Extract File Features), UC04 (Perform Pattern Matching)

**Special Requirements:**

1. System must support Unicode paths and long filenames.
2. Traversal must be efficient even for directories with large amount of files

**Assumptions:**

1. User provides a readable and accessible folder path.
2. Directory contains files compatible with the system's analysis capabilities.

**Notes and Issues:**

* Deep recursion can be resource intensive.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC03 | | |
| Use Case Name: | Extract File Features | | |
| Created By: | Yameen | Last Updated By: | Yameen |
| Date Created: | 1st May 2025 | Date Last Updated: | 1st May 2025 |

**Use Case 3:** UC03 – Extract File Features

**Actor:** System

**Description:** System reads metadata such as file size, extension, and timestamps to aid in detection analysis.

**Preconditions:**

1. File list is generated successfully by UC02 (Scan Files).
2. Files are accessible and readable by the system.

**Postconditions:**

* 1. Extracted metadata is stored or passed to downstream modules for further analysis.
  2. Data becomes available to UC04 (Pattern Matching) and UC13 (AI Classification).

**Priority:** High

**Frequency of Use:** Every time a file is scanned.

**Normal Course of Events:**

1. System accesses each file from the queue generated in UC02.
2. Retrieves metadata including:

* File size (in bytes)
* File extension/type
* Last modified date/time
* Creation date (if available)

1. Stores or forwards this information for:

* Signature-based rules.
* Behavioural heuristics.
* Machine learning input

1. Proceeds to UC04 and UC13.

**Alternative Courses:**

* **UC03.AC.1:** For archived or compound files only top-level metadata is extracted.

**Exceptions:**

* **UC03. EX.1:** File is corrupted or locked, system logs an error and skips to next file.
* **UC03. EX.2:** Metadata API call fails due to system restrictions (e.g., sandboxed OS).

**Includes:** UC04 and UC13.

**Special Requirements:**

1. System must handle both small and large files efficiently.
2. File path and metadata must be stored in memory or temp storage securely.
3. Should support both common and uncommon file formats.

**Assumptions:**

1. Files reside on a local disk.
2. Metadata fields are available through standard OS calls.

**Notes and Issues:**

* Some older or binary file types may have incomplete or misleading metadata.
* Metadata integrity cannot be guaranteed for obfuscated/masked malware samples.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **UC04** | | |
| Use Case Name: | **Pattern Matching** | | |
| Created By: | Usman | Last Updated By: | Usman |
| Date Created: | 1st May 2025 | Date Last Updated: | 1st May 2025 |

**Use Case 4: UC04 – Pattern Matching**

**Actor:** System

**Description:** The system reads file contents and matches them against known malware patterns stored in the local signature database.

**Preconditions:**

1. Metadata has been extracted for each file by UC03.
2. The malware signature database is available and loaded into memory.
3. File is readable and not encrypted or binary incompatible.

**Postconditions:**

1. Files that match any known malware signature pattern are flagged.
2. Matching results are passed to UC06 (Detect Malicious Files) for action.

**Priority:** High

**Frequency of Use:** Once per file, during every scan session.

**Normal Course of Events:**

1. The system loads the signature database into memory.
2. Reads content from each file passed from UC03.
3. Applies each pattern to the file contents.
4. If a match is found, the file is marked as "Matched" and passed to UC06.
5. If no match is found, file is labelled "Clean" (passed to AI module).

**Alternative Courses:**

* **UC04.AC.1:** Partial match detected; system may apply a heuristic threshold.
* **UC04.AC.2:** Pattern contains deprecated syntax, skipped and reported during load.

**Exceptions:**

* **UC04.EX.1:** Signature database is outdated; system logs a warning but continues.
* **UC04.EX.2:** File content cannot be read (e.g., encoding mismatch); log and skip.

**Includes:**

1. UC03 - Extract File Features
2. Supports UC06 - Detect Malicious Files

**Special Requirements:**

1. Pattern matcher must support wildcard entries.
2. Matching must maintain performance standards.
3. Patterns should be loaded securely and validated before execution to avoid injection.

**Assumptions:**

1. Signature database is curated and updated regularly.
2. Files are in a readable format (not encrypted or compressed at this stage).

**Notes and Issues:**

* Regex-heavy patterns may lead to false positives if not optimized.
* Pattern engine must be robust enough to handle thousands of entries efficiently.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **UC05** | | |
| Use Case Name: | **Generate File Hash** | | |
| Created By: | Daniyal | Last Updated By: | Daniyal |
| Date Created: | 1st May 2025 | Date Last Updated: | 1st May 2025 |

**Use Case 5: UC05 – Generate File Hash**

**Actor:** System

**Description:** Computes a SHA-256 cryptographic hash of each file to serve as a unique identifier for logging, audit trails, or potential threat lookup.

**Preconditions:**

1. File list is generated by UC02.
2. Files are accessible for read operations.
3. OpenSSL or an equivalent hashing library is available in the system.

**Postconditions:**

1. SHA-256 hashes are successfully generated.
2. Hashes are written to logs in UC07 for every scanned file.

**Priority:** Medium

**Frequency of Use:** Once per file scanned.

**Normal Course of Events:**

1. System opens each file for scan queue
2. read file content as byte stream.
3. Computes SHA-256 hash using the system's hashing module.
4. Passes hash result to UC07 (Log Scan Results) and optionally UC11 (API-Based Threat Scan)
5. Continues to next file.

**Alternative Courses:**

* **UC05.AC.1:** File is zero-byte or empty; system logs hash as predefined empty value.

**Exceptions:**

* **UC05. EX.1:** File cannot be read (due to permission or corruption); system skips hash generation and logs warning.
* **UC05. EX.2:** Hashing library fails to initialize; system logs critical error and halts hashing component.

**Includes:** UC02 – Scan Files

**Special Requirements:**

1. Hashing function must be memory-efficient and fast.
2. Should produce results in consistent 64-character hexadecimal format.
3. File content must not be altered during hashing.

**Assumptions:**

1. The file remains unchanged during hashing.
2. Hash values may be reused for identification in AI and API modules.

**Notes and Issues:**

* Logging accuracy depends on correct pairing of file path and its hash.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **UC06** | | |
| Use Case Name: | **Detect Malicious Files** | | |
| Created By: | Usman | Last Updated By: | Usman |
| Date Created: | 2nd May 2025 | Date Last Updated: | 2nd May 2025 |

**Use Case 6: UC06 – Detect Malicious Files**

**Actor:** System

**Description:** The system evaluates the result of pattern matching to decide whether a file is malicious, clean, or suspicious. This decision guides alerting, logging, and quarantine actions.

**Preconditions:**

1. Pattern matching has been performed on the file by UC04.
2. The file’s status and metadata are available for analysis.
3. Rules or thresholds for detection are defined in the system.

**Postconditions:**

1. Files are labelled (malicious / clean / suspicious).
2. Malicious files are passed to UC07 (Log Results) and UC08 (Alert User).
3. Optionally passed to UC09 or UC10 (Quarantine or Delete).

**Priority:** High

**Frequency of Use:** Every scanned file that reaches detection stage.

**Normal Course of Events:**

1. System receives match status from UC04.
2. Applies detection logic or rule-based decision:

* Full match → flag as malicious.
* No match → flag as clean.
* Partial/heuristic match → flag as suspicious.

1. Flags the file accordingly.
2. Triggers UC07 to record the result.
3. Triggers UC08 to alert the user if malicious or suspicious.

**Alternative Courses:**

* **UC06.AC.1:** Detection engine uncertain; file marked as "Unknown", log with warning.
* **UC06.AC.2:** Detection result deferred for further review or AI validation.

**Exceptions:**

* **UC06. EX.1:** Corrupted match result; log error and skip detection for that file.
* **UC06. EX.2:** No valid pattern rules found; system defaults to clean and logs a warning.

**Includes:**

1. UC04 – Perform Pattern Matching
2. Triggers: UC07 – Log Results, UC08 – Alert User

**Special Requirements:**

1. Detection decisions must occur in milli seconds
2. System must be able to differentiate between false positives and actual threats.
3. Rule-based thresholds should be configurable.

**Assumptions:**

1. The signature database contains trustworthy rules.
2. Detection logic is reviewed and tested.
3. Hash- or API-based verification may override initial decision.

**Notes and Issues:**

* Over-aggressive detection logic can lead to false positives.
* Missed detections (false negatives) must be minimized through rule tuning.
* Modular detection engine allows for AI integration in without changing architecture.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **UC07** | | |
| Use Case Name: | **Log Scan Results** | | |
| Created By: | Daniyal | Last Updated By: | Daniyal |
| Date Created: | 2nd May 2025 | Date Last Updated: | 2nd May 2025 |

**Use Case 7: UC07 – Log Scan Results**

**Actor:** System

**Description:** The system saves the results of the malware scan—including file statuses, hash values, and detection outcomes—into a structured log file in formats such as JSON or CSV. These logs support further analysis, auditing, and user review.

**Preconditions:**

1. File scan and malware detection processes (UC06) have been completed.
2. Directory and files have been successfully traversed (UC02).

**Postconditions:**

1. A structured log file is created in the predefined log directory.
2. Each scanned file is represented with its metadata, hash, and scan status.

**Priority:** High

**Frequency of Use:** Once per scan session.

**Normal Course of Events:**

1. The system gathers all processed file metadata and detection results.
2. Formats data according to selected output type.
3. Write the formatted data to a log file on disk.
4. Confirms successful writing operations.

**Alternative Courses:**

* **UC07.AC.1:** User selects a custom format; system exports accordingly.
* **UC07.AC.2:** Multiple output formats required; system generates both JSON and CSV.

**Exceptions:**

* **UC07.EX.1:** Disk full or permission denied; log fails, and user is alerted.
* **UC07.EX.2:** File path invalid; system attempts alternate default path and logs fallback.

**Includes:**

1. UC06 – Detect Malicious Files
2. Supports: UC08 – Alert User

**Special Requirements:**

1. Log structure must support sorting/filtering.
2. Each log file must be timestamped for version control.
3. Sensitive log access should be restricted (e.g., admin-only mode).

**Assumptions:**

1. Log directory is preconfigured and accessible.
2. Encryption is applied for secure environments.

**Notes and Issues:**

* Log retention policy should be defined to prevent storage bloat.
* JSON preferred for compatibility with visualization/analysis tools.



**Use Case 8: UC08 – Alert User**

**Actor:** System

**Description:** The system alerts the user in real time by displaying information about detected malicious files directly in the terminal or GUI. Alerts include filenames, threat types, and recommended actions.

**Preconditions:**

1. Malicious files have been identified during pattern matching and classification (UC06).
2. Scan results are available and correctly formatted.

**Postconditions:**

1. The user is made aware of detected threats via visual or terminal alerts.
2. Alert data is optionally logged for audit and post-analysis (linked to UC07).

**Priority:** High

**Frequency of Use:** Every time threats are detected**.**

**Normal Course of Events:**

1. System processes results from UC06.
2. System constructs readable alert messages.
3. Messages are displayed in the terminal or UI panel, showing:
   * File name
   * Threat type or signature
   * Suggested action (e.g., quarantine, delete)

**Alternative Courses:**

* **UC08.AC.1:** User configures silent mode; alerts are suppressed but logged.
* **UC08.AC.2:** If terminal-based UI is replaced with graphical interface, pop-up dialog boxes may be used

**Exceptions:**

* **UC08.EX.1:** Output interface failure; fallback alert to log file (UC07).
* **UC08.EX.2:** Incomplete or corrupt data from UC06 ; display warning and skip alert.

**Includes:**

UC06 – Detect Malicious Files  
Supports: UC07 – Log Scan Results

**Special Requirements:**

1. Alerts must be clear, non-technical for general users.
2. Colours can improve readability.
3. No sensitive personal data should be exposed in alerts.

**Assumptions:**

1. User is present and actively monitoring scan progress.
2. Terminal or UI session is active during scan process.

**Notes and Issues:**

* In future versions (FYP 2), integrate sound alerts, email notifications, or system tray pop-ups.
* Alert throttling might be required for large-scale scans to avoid flooding.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **UC09** | | |
| Use Case Name: | **Quarantine Malware** | | |
| Created By: | Usman | Last Updated By: | Usman |
| Date Created: | 3rd May 2025 | Date Last Updated: | 3rd May 2025 |

**Use Case 9: UC09 – Quarantine Malware**

**Actor:** System/User

**Description:** The system provides an option to isolate detected malicious files by moving them into a secured, access-restricted quarantine folder, preventing execution or interaction.

**Preconditions:**

1. Malicious files have been flagged by UC06 – Detect Malicious Files.
2. Quarantine feature is enabled and configured.
3. The user has permissions to modify file locations.

**Postconditions:**

1. Malicious files are no longer accessible from their original paths.
2. Files are secured within a quarantine folder with limited access.
3. System logs the quarantine action (linked to UC07 – Log Scan Results).

**Priority:** High

**Frequency of Use:** Every time threats are found, and quarantine is selected.

**Normal Course of Events:**

1. System lists detected threats for user review.
2. Prompts the user to approve quarantine action.
3. Upon confirmation:

* A quarantine folder is created if it doesn’t exist.
* Files are moved securely into this folder.
* File permissions are altered to prevent execution.

1. Triggers UC07 to update logs and UC08 if alerts need updating.

**Alternative Courses:**

* **UC09.AC.1:** User enables auto-quarantine; system skips prompt and proceeds directly.
* **UC09.AC.2:** User rejects quarantine; system offers “Ignore” or “Delete” options.

**Exceptions:**

* **UC09.EX.1:** Insufficient permissions; system logs failure and notifies the user.
* **UC09.EX.2:** File in use or locked; system retries or prompts for action.
* **UC09.EX.3:** Quarantine folder path not writable; fallback to alternative secure location or notify admin

**Includes:**

1. UC07 – Log Scan Results
2. UC08 – Alert User

**Special Requirements:**

1. Quarantine folder must have restricted access.
2. File renaming may be used to prevent accidental execution.
3. Ensure data is recoverable or deletable later.

**Assumptions:**

1. User understands the purpose of quarantining.
2. System is not running with restricted privileges.

**Notes and Issues:**

* May need to include quarantine management GUI, restore options, and automated cleanup.
* integrating antivirus database checks to re-verify quarantined files.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **UC10** | | |
| Use Case Name: | **Remove Malware** | | |
| Created By: | Usman | Last Updated By: | Usman |
| Date Created: | 3rd May 2025 | Date Last Updated: | 3rd May 2025 |

**Use Case 10: UC10 – Remove Malware**

**Actor:** User

**Description:** The user permanently deletes a file that has previously been identified as malicious and was placed into quarantine.

**Preconditions:**

1. The file must have already been quarantined in **UC09 – Quarantine Malware**.
2. The antivirus application has the necessary permissions to delete files from the quarantine folder.
3. User has access to the quarantine management interface (CLI or GUI).

**Postconditions:**

1. The file is completely and irreversibly removed from the system.
2. Deletion is logged as part of the scan history.
3. System notifies the user of successful deletion.

**Priority:**

Medium (dependent on user action and threat level)

**Frequency of Use:**

Periodically, based on user’s review of quarantined files.

**Normal Course of Events:**

1. User navigates to the quarantine section of the antivirus.
2. User selects one or more files to delete.
3. System prompts for confirmation.
4. Upon confirmation:
   * Files are securely deleted from storage.
   * Logs are updated (linked to UC07 – Log Scan Results).
   * Alerts are cleared or updated (linked to UC08 – Alert User).

**Alternative Courses:**

* **UC10.AC.1:** User cancels deletion after confirmation prompt; no action taken.
* **UC10.AC.2:** User opts to “Restore” instead of delete; system triggers restore flow.

**Exceptions:**

* **UC10.EX.1:** File already deleted or missing; system logs inconsistency and informs user.
* **UC10.EX.2:** File in use or locked; deletion fails, error logged, user notified.
* **UC10.EX.3:** Deletion permissions denied; system requests elevated access or aborts with warning.

**Includes:**

1. UC07 – Log Scan Results
2. UC08 – Alert User

**Special Requirements:**

1. Secure deletion method to prevent file recovery.
2. Prompt confirmation dialogue to prevent accidental data loss.
3. Deleted file should not remain in system trash/recycle bin.

**Assumptions:**

1. User understands deletion is permanent.
2. Antivirus has elevated or administrative rights if required.

**Notes and Issues:**

* Option to batch-delete or schedule cleanup may be added in later versions.
* Future enhancements can log deleted file hashes for historical traceability.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **UC11** | | |
| Use Case Name: | **API-Based Threat Scan** | | |
| Created By: | Daniyal | Last Updated By: | Daniyal |
| Date Created: | 3rd May 2025 | Date Last Updated: | 3rd May 2025 |

**Use Case 11: UC11 – API-Based Threat Scan**

**Actor:** System/User

**Description:** The system or user via interface submits the SHA-256 hash of a file to an external threat intelligence API to check for known threats.

**Preconditions:**

1. SHA-256 hash of the file must have been successfully generated in **UC05 – Generate File Hash**.
2. Internet connectivity must be available.
3. API key must be valid and the service reachable.
4. The file hash must not have been previously submitted in the current scan session.

**Postconditions:**

1. API response is stored for each hash in the local result logs.
2. If a threat is reported by the API, the file is flagged and sent to **UC06 – Detect Malicious Files**.
3. System logs both the query and response status.

**Priority:** High

**Frequency of Use:** For every file hash generated during scans if enabled/configured.

**Normal Course of Events:**

1. System collects hash generated from UC05.
2. System formats and sends an HTTPS request to the external API with the hash.
3. API responds with a threat intelligence report.
4. System parses response:
   * If malicious: flags file, triggers **UC06** and **UC08**.
   * If clean: logs the result.
5. Results are saved and linked to **UC07 – Log Scan Results**.

**Alternative Courses:**

1. **UC11.AC.1:** API returns ambiguous or unknown status; System marks as “Unknown” and logs for review.
2. **UC11.AC.2:** User is prompted to approve submission.

**Exceptions:**

1. **UC11.EX.1:** API unreachable due to network failure or server-side issue; system retries up to 3 times, then logs the failure and optionally alerts the user.
2. **UC11.EX.2:** API quota exceeded or key invalid; aborts submission and informs user.

**Includes:**

1. UC06 – Detect Malicious Files
2. UC07 – Log Scan Results
3. UC08 – Alert User

**Special Requirements:**

1. Secure HTTPS communication with API endpoints.
2. Configurable API keys and retry logic.
3. Caching of past results to avoid duplicate queries.

**Assumptions:**

1. User/system has access to a supported threat intelligence API.
2. File hashes are valid and supported by the API.

**Notes and Issues:**

* May add support for multiple APIs (e.g., hybrid detection).
* Potential privacy concern requires policy handling for external submissions.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **UC12** | | |
| Use Case Name: | **Classify Files with AI** | | |
| Created By: | Yameen | Last Updated By: | Yameen |
| Date Created: | 3rd May 2025 | Date Last Updated: | 3rd May 2025 |

**Use Case 12: UC12 – Classify Files with AI**

**Actor:** System

**Description:** The system uses a pre-trained machine learning model to analyze file metadata and classify files as **Safe** or **Suspicious**.

**Preconditions:**

1. File metadata must be successfully extracted in **UC03 – Extract File Features**.
2. A trained ML model must be available and loaded into memory.
3. Required ML libraries and runtime environment must be configured properly.
4. Input data must be pre-processed into the format expected by the ML model.

**Postconditions:**

1. Each file is assigned a classification label ("Safe" or "Suspicious").
2. Classification results are passed to **UC06 – Detect Malicious Files** and logged via **UC07 – Log Scan Results**.
3. If Suspicious; system triggers **UC08 – Alert User** and optionally **UC09 – Quarantine Malware**.

**Priority:** Medium-High

**Frequency of Use:** Used during each scan where ML classification is enabled.

**Normal Course of Events:**

1. System receives metadata for each file from UC03.
2. Metadata is formatted and passed to the AI model.
3. Model returns a prediction label.
4. System stores classification result.
5. Suspicious classifications proceed to UC06, UC08, and UC09 if necessary.

**Alternative Courses:**

* **UC12.AC.1:** If model confidence is low (below threshold), system flags the file as "Needs Manual Review".

**Exceptions:**

* **UC12.EX.1:** ML model fails to load; scan continues without AI classification.
* **UC12.EX.2:** Model input shape mismatch or data error; skip classification for affected file and log error

**Includes:**

1. UC06 – Detect Malicious Files
2. UC07 – Log Scan Results
3. UC08 – Alert User
4. UC09 – Quarantine Malware (conditionally)

**Special Requirements:**

1. Model must be validated for accuracy, precision, and recall.
2. Requires sufficient compute resources for inference (CPU/GPU).
3. Real-time prediction performance optimization.

**Assumptions:**

1. Model is trained and periodically updated with recent malware data.
2. AI model operates in a secure and isolated inference environment.
3. AI-based classification is supplementary and not the sole detection method.

**Notes and Issues:**

* A future enhancement may involve retraining the model with feedback from quarantined files.
* If false positives are high, a confidence threshold adjustment may be necessary.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **UC13** | | |
| Use Case Name: | **Update Signature Database** | | |
| Created By: | Usman | Last Updated By: | Usman |
| Date Created: | 4th May 2025 | Date Last Updated: | 4th May 2025 |

**Use Case 13: UC13 – Update Signature Database**

**Actor:** User/System

**Description:** Updates the local signature database with the latest known malware signatures, either through scheduled automatic updates or on user command.

**Preconditions:**

1. Internet connection must be available for fetching updates.
2. Existing signature database must be in a writable state.
3. If system-driven, the update scheduler must be active.

**Postconditions:**

1. Signature database is updated with the latest signatures.
2. A log entry is created indicating the update status (success/failure).
3. Updated signatures are immediately available for UC04 (Pattern Matching).

**Priority:** High

**Frequency of Use:** AutomaticallyorOn-demand by the user.

**Normal Course of Events:**

1. System (or user) initiates the signature update routine.
2. System connects to a trusted update server.
3. Server responds with update package.
4. System downloads and verifies integrity.
5. New signature file replaces or merges with the local database.
6. Success status is logged.

**Alternative Courses:**

* **UC13.AC.1:** User cancels update before completion; retain old signature set and log the action.

**Exceptions:**

1. **UC13.EX.1:** Update server unreachable; log error and notify user.
2. **UC13.EX.2:** Signature file integrity check fails; discard update and raise warning.
3. **UC13.EX.3:** File system permission error prevents update; notify user and abort.

**Includes:**

None directly but impacts UC04 – Perform Pattern Matching and UC06 – Detect Malicious Files.

**Special Requirements:**

1. Secure connection (HTTPS) to update server.
2. Verification of update file (e.g., SHA-256 checksum or digital signature).
3. Graceful fallback to previous version in case of failure.

**Assumptions:**

1. Update server is trusted and maintained.
2. Signature files are compatible with local pattern matching engine.
3. User/system has permission to overwrite database files.

**Notes and Issues:**

* Signature updates should not interrupt ongoing scans.
* A future improvement may involve differential updates for optimization

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **UC14** | | |
| Use Case Name: | **View Scan Report** | | |
| Created By: | Daniyal | Last Updated By: | Daniyal |
| Date Created: | 4th May 2025 | Date Last Updated: | 4th May 2025 |

**Use Case 14: UC14 – View Scan Report**

**Actor:** User

**Description:** Allows the user to access and review previously generated scan reports from the system logs.

**Preconditions:**

1. Logs must be available, previously created through UC07 – Log Scan Results.
2. Application is running with access to log directory.

**Postconditions:**

1. Scan reports are displayed to the user.
2. User can optionally export reports in different formats.

**Priority: Medium** – Enhances usability and transparency, but not essential for detection.

**Frequency of Use:** Occasional – typically after a scan or when reviewing historical data.

**Normal Course of Events:**

1. User navigates to the scan history or report section via UI or command.
2. System fetches available scan logs from the storage location.
3. User selects filters such as date, file path, or threat type.
4. System renders the filtered results in an interactive or readable format.
5. User optionally exports the report in JSON/CSV format.

**Alternative Courses:**

* **UC14.AC.1:** User opens a specific log file directly from the file browser instead of via app interface.

**Exceptions:**

1. **UC14.EX.1:** Log file missing or corrupted; system displays error and suggests a rescan.
2. **UC14.EX.2:** File permissions denied for log access; system alerts the user and exits the report view.

**Includes:**

May use shared UI components or file readers common to UC07 – Log Scan Results.

**Special Requirements:**

1. Export options for report in CSV and JSON formats.
2. Sort and search capabilities for enhanced analysis.
3. Scrollable and paginated display if logs are extensive.

**Assumptions:**

1. User has permission to view logs.
2. Logs are structured in a readable and parseable format.
3. Storage location has not changed.

**Notes and Issues:**

* Need an auto-archive logger.