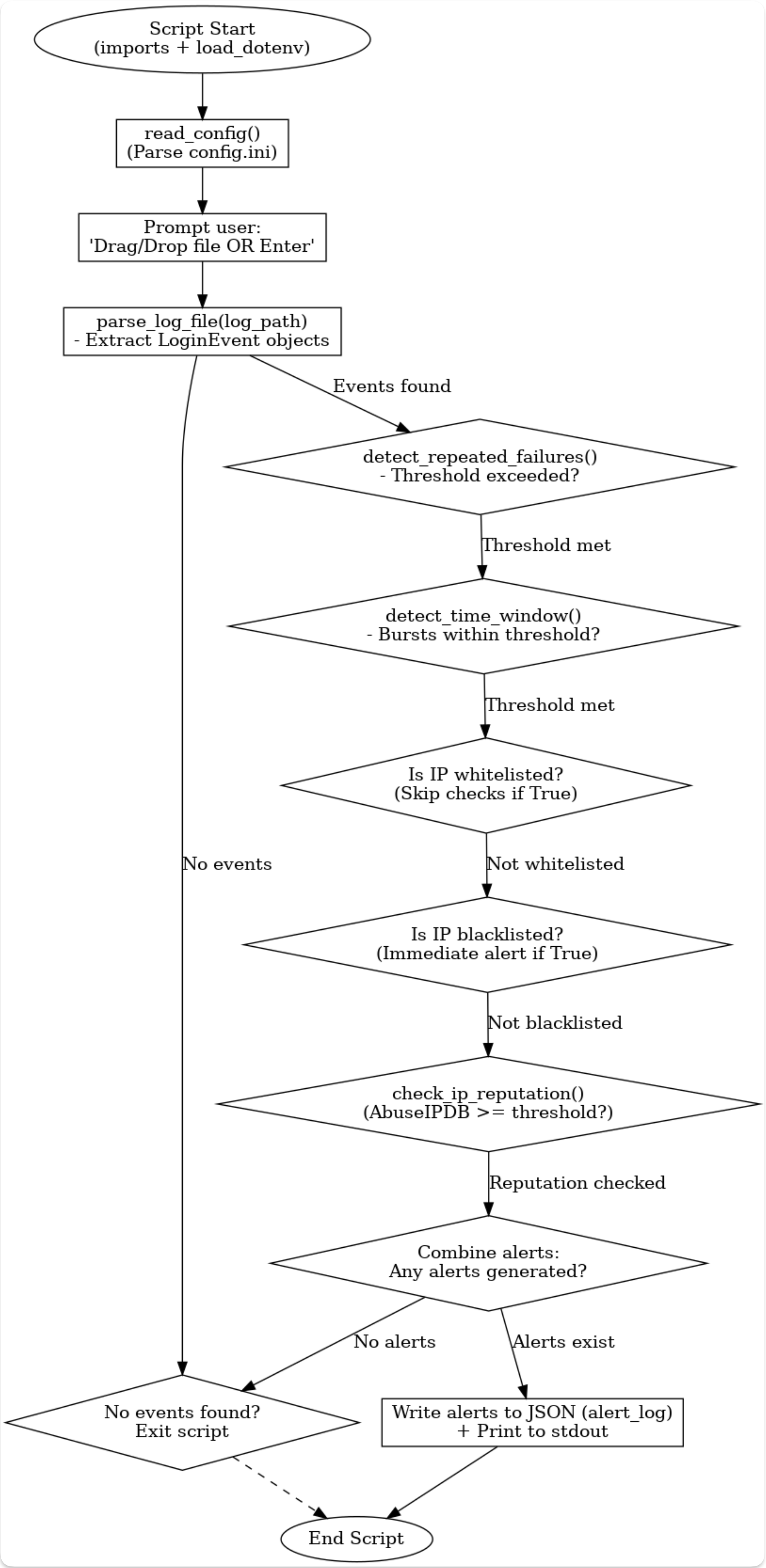
This script monitors SSH login attempts, identifies suspicious patterns, checks IP reputations, and generates alerts. Here's a breakdown of its functionality, following the order of execution:

1. **Load Environment Variables:**
   - The script starts by loading environment variables from a `.env` file using `load_dotenv()`. This is primarily to fetch the `ABUSEIPDB_API_KEY` if it exists.
2. **Configuration Loading (read_config)**
   - **If `config.ini` exists and is readable:** Default paths, thresholds, and whitelist/blacklist entries are loaded.
   - **If `config.ini` is missing or unreadable:** Fallback values (hardcoded defaults) are used.
3. **User Input for Log File Path**
   - The script prompts the user to enter a log file path (drag-and-drop or manual entry).
   - **If the user provides input:** The path is sanitized (quotes removed).
   - **If the user presses Enter without input:** The default path from `config.ini` (or its fallback) is used.
4. **Log File Parsing (parse_log_file)**
   - **If the log file is accessible:** It's read line by line.
   - **If the log file is not found or inaccessible:** An error is printed, and the script exits.
   - **Event Classification:**
     - **Failed Password Events:** Lines matching `"Failed password for .* from ([^\s]+) port"` are identified. The IP address is extracted. A `FAILED` event (with timestamp, IP, raw line) is created and added to the `events` list.
     - **Successful Login Events:** Lines matching `"Accepted (?:password|publickey) for .* from ([^\s]+) port"` are identified. The IP address is extracted. A `SUCCESS` event (with timestamp, IP, raw line) is created and added to the `events` list.
     - **No IP Validation at Parse Time:** Trying to keep it simple. Future roadmap item.

5. **Early Exit if No Events:**
   - **If no events were parsed:** A message is printed, and the script exits.



6. **Pattern Detection:**
   - **Detect Repeated Failures (detect_repeated_failures):**
     - The `events` list is sorted by IP and then timestamp.
     - The function iterates through the sorted events, tracking consecutive failures for each IP.
     - **Alert Condition:** If the number of consecutive failed attempts for an IP exceeds `failed_threshold` (either followed by a success or at the end of the log), an alert is generated.
     - **Severity:** Determined by `determine_severity_by_count` (HIGH, MEDIUM, or LOW based on the failure count).
   - **Detect Time Window (detect_time_window):**
     - The `events` list is sorted by timestamp.
     - The function iterates through `FAILED` events, checking for bursts within a `time_window_seconds` window.
     - **Alert Condition:** If failures within the time window for an IP exceed `time_window_threshold`, an alert is generated.
     - **Severity:** Determined by `determine_severity_by_count`.

7. **Unique IP Collection**
   - A set of unique IPs (`unique_ips`) is created from all parsed events.

8. **IP Checks (Whitelist, Blacklist, Reputation)**
   - The script iterates through each `ip` in `unique_ips`.
   - **Whitelist Check (is_whitelisted):**
     - If the IP is in the whitelist (supports individual IPs or CIDR), it's skipped for further processing.
   - **Blacklist Check (is_blacklisted):**
     - If the IP is in the blacklist (supports individual IPs or CIDR), a `HIGH` severity alert is immediately generated and added to `rep_alerts`.
   - **Reputation Check (check_ip_reputation):**
     - If the IP is not whitelisted or blacklisted:
       - The AbuseIPDB API is queried (if an API key is available).
       - **If the reputation score >= `reputation_threshold`:** A `HIGH` severity alert is generated and added to `rep_alerts`.
       - **If the reputation score < `reputation_threshold`:** No alert is created.

9. **Alert Combination**
   - Alerts from `detect_repeated_failures`, `detect_time_window`, and `rep_alerts` are combined into `all_alerts`.

10. **Early Exit if No Alerts:**
    - **If `all_alerts` is empty:** A message is printed, and the script exits.

11. **Final Output**
    - **Write to JSON File:**
      - Alerts in `all_alerts` are written to the JSON file specified by `alert_log` (or the default).
      - Success or failure is reported to the console.
    - **Print to Console:**
      - All alerts in `all_alerts` are printed to the console in JSON format.
      - The total number of alerts is printed.