Apache Log Analysis and Security Reporting Script

1. Overview

This Python script is designed to analyze Apache web server logs for potential security threats. It parses log data, extracts key information, detects suspicious behaviors (like brute force or scanning attempts), and generates summary reports and visualizations.

2. Dependencies

The script uses the following Python libraries:

- pandas – for data manipulation.

- re – for regex pattern matching.

- csv – for reading CSV log files.

- matplotlib.pyplot – for data visualization.

- urllib.parse – to extract URL components.

- collections.Counter – for frequency analysis.

- warnings – to suppress unnecessary warnings.

3. Secure Log Reader

Function: secure_log_reader(log_file_path)

- Reads a CSV-formatted Apache log file.
- Extracts and organizes log lines into log_data and agent_info.
- Returns a Pandas DataFrame.

4. Apache Log Processor

Function: process_apache_logs(input_log_file)

- Uses regex to extract structured fields like client_ip, http_method, request_url, response_code, etc.
- Converts time strings to datetime format.
- Parses URLs to separate hostname and uri_path.
- Returns a cleaned and structured DataFrame.

5. Threat Detection Modules

Brute Force Detector: identify_bruteforce(log_data, attempt_threshold=10)

- Detects IPs that repeatedly failed authentication (401/403 responses).
- Returns a list of suspicious IPs exceeding a threshold.

Scanner Detector: identify_scanners(log_data, scan_threshold=15)

- Identifies IPs with abnormally high request volume.
- Flags these as potential scanners or bots.

User Agent Analyzer: examine_user_agents(log_data)

- Checks for known hacking tools in the User-Agent header.
- Flags log entries with tools like sqlmap, nikto, burp, etc.

6. Visualization

Function: create_visuals(log_data)

- Bar chart of HTTP response codes (response_codes.png).
- Top 15 client IPs by frequency (top_clients.png).

7. Reporting

Function: create_analysis_report(log_data, report_file='security_analysis.txt')

- Generates a text report including:
  - Time range of logs.
  - Brute-force IPs.
  - Scanning IPs.
  - Suspicious user agents.
  - Top client IPs and requested URLs.

8. Main Execution Block

Function: if _name_ == "_main_": (Should be corrected to if __name__ == "__main__":)

- Triggers the full workflow:
  - Parses the log file.
  - Runs analysis and visualization.
  - Saves outputs: .txt, .csv, and .png files.

9. Output Files

- security_analysis.txt – Full text report.
- response_codes.png – HTTP response code distribution.
- top_clients.png – Top IPs making requests.
- processed_apache_logs.csv – Cleaned data for further analysis.

10. Final Notes

- Ensure the input file apache_logs.csv exists in the same directory.
- Adjust thresholds in identify_bruteforce and identify_scanners for different sensitivity.
- Correct the main block condition: Replace _name_ with __name__.

11. Code Explanation

This section provides a detailed explanation of how the code works, step by step:

1. The script starts by importing necessary libraries like pandas for data manipulation, re for regex pattern matching, csv for reading log files, and matplotlib for plotting charts.

2. The `secure_log_reader` function reads a CSV file containing Apache log entries. It handles encoding issues and splits each log entry into two parts: the request log data and the user agent string. These are returned in a DataFrame.

3. The `process_apache_logs` function takes the logs, uses a regex pattern to extract useful fields such as the client's IP, timestamp, HTTP method, requested URL, response code, and more. It also parses and cleans the data, converting types and extracting components from URLs.

4. Security analysis functions are provided:
   - `identify_bruteforce` finds IPs with many failed login attempts (401/403 errors).
   - `identify_scanners` identifies IPs with high numbers of requests.
   - `examine_user_agents` flags requests made with known malicious tools (e.g., sqlmap, nikto).

5. `create_visuals` generates visual summaries of the logs, including a bar chart of response codes and a chart of the most active IP addresses.

6. The `create_analysis_report` function creates a text file summarizing the log analysis. It includes time ranges, detected threats, top client IPs, and frequent URLs.

7. The main execution block (`if __name__ == "__main__":`) coordinates everything: it reads logs, performs analysis, saves visualizations, and writes reports. This block will only execute when the script is run directly (not imported as a module). Note: the original script mistakenly used `_name_` instead of `__name__`, which needs correction.