AI Generated pseudocodes

Algorithm 1. Generating Adversarial Log Data with ChatGPT

Input: log_data (a list of log entries)
Output: report, alerts, response_actions for real-time detection

1: Function **preprocess_data**(log_data)
2: Initialize processed_data as an empty list
3: For each entry in log_data
4: Create processed_entry with timestamp, user, action, and ip_address from entry
5: Append processed_entry to processed_data
6: Return processed_data

7: Function **analyze_data**(processed_data, ai_model)
8: Initialize analysis_results with empty lists for threats_detected and anomalies
9: For each entry in processed_data
10: Use ai_model to assess the risk level of each log entry
11: If ai_model flags entry as high risk or potential threat
12: Append entry to analysis_results['threats_detected']
13: Else if ai_model flags entry as anomaly
14: Append entry to analysis_results['anomalies']
15: Else
16: Continue to next entry
17: Return analysis_results

18: Function **generate_report**(analysis_results)
19: Initialize report as a string summarizing the threats detected
20: For each threat in analysis_results['threats_detected']
21: Append threat details to report
22: Return report

23: Function **real_time_threat_detection**(splunk_service, chatgpt_api_key, real_time_data_stream)
24: Initialize alerts and response_actions as empty lists
25: For each data in real_time_data_stream
26: Analyze data and generate alerts
27: Return alerts, response_actions

28: Main Program
29: Call preprocess_data with log_data
30: Call analyze_data with processed_data and 'ai_model_placeholder'
31: Call generate_report with analysis_results
32: Print "Analysis Report:" and report

For real-time threat detection, initialize Splunk service and ChatGPT API key, and get real-time data stream). Call real_time_threat_detection with splunk_service, chatgpt_api_key, and real_time_data_stream).

Algorithm 2. AI-Driven Test Case Generation

Input: Number of test cases (num_tests), ChatGPT API key (chatgpt_api_key), Splunk service connection details (splunk_service)
Output: Test case analysis results

1: Function **generate_adversarial_log_data**(chatgpt_api_key)
2: Set prompt to "Generate a complex cybersecurity log entry."
3: Set headers for API request with Authorization as "Bearer" plus chatgpt_api_key
4: Send POST request to ChatGPT-4-Turbo API with prompt and max_tokens
5: Set log_data to the response text from API
6: Return log_data

7: Function **ingest_log_data_to_splunk**(log_data, splunk_service)
8: Ingest log_data into Splunk using HTTP Event Collector
9: Print "Log data ingested to Splunk."

10: Function **run_splunk_search_query**(splunk_service, query)
11: Create a Splunk search job with the given query
12: Wait until the job is ready
13: Retrieve and return the search results in JSON format

14: Function **create_and_analyze_test_cases**(num_tests, chatgpt_api_key, splunk_service)
15: For each test case from 1 to num_tests
16: Generate adversarial log data using generate_adversarial_log_data
17: Ingest the log data into Splunk using ingest_log_data_to_splunk
18: Define a Splunk search query for analysis
19: Run the query using run_splunk_search_query
20: Print the log data and Splunk analysis results

21: Main Program
22: Set chatgpt_api_key, splunk_host, splunk_port, splunk_username, and splunk_password
23: Connect to Splunk service
24: Create and analyze num_tests test cases using ChatGPT and Splunk