Assignment-4

Name: Darshan Bele
Roll No: 14110
Class: BE – A – A1

```
import argparse
import json
import re import
os
from datetime import datetime, timedelta

def parse_log_file(file_path, source_config):
    """
    Parses a single log file based on its configuration.

    Args:
        file_path (str): The path to the log file.         source_config
(dict): The configuration for this log source.

    Returns:
        list: A list of structured log event dictionaries.
    """
    parsed_events = []
    log_pattern = re.compile(source_config['regex'])
current_year = datetime.now().year

    print(f"[+] Parsing {file_path}...")     try:
with open(file_path, 'r') as f:             for
line_num, line in enumerate(f, 1):
match = log_pattern.match(line)
if match:
                    event_data = match.groupdict()
                    event_data['log_type'] = source_config['type']
event_data['raw_message'] = line.strip()

                    # Handle timestamp parsing
ts_str = event_data.get('timestamp')
                    ts_format = source_config['timestamp_format']

try:
                        # For formats without a year, add the current year
if '%Y' not in ts_format:
                            dt_obj = datetime.strptime(f"{current_year} {ts_str}", f"%Y
{ts_format}")                   else:
                            # For formats with timezone, handle it
                            if '%' in ts_format[-1]: # Simple check for timezone
directive                         ts_str = ts_str[:-6] + ts_str[-5:].replace(':', '')
ts_format = ts_format.replace('%z', '%z')                       dt_obj =
datetime.strptime(ts_str, ts_format)                    else:
                                dt_obj = datetime.strptime(ts_str, ts_format)
                    event_data['timestamp_dt']          =
dt_obj
parsed_events.append(event_data)
except ValueError as e:
                    print(f"  [!] Warning: Could not parse timestamp on line {line_num} in
{file_path}. Error: {e}")
             else:
                 print(f"  [!] Warning: Line {line_num} in {file_path} did not match regex.")
except FileNotFoundError:
      print(f"  [-] Error: Log file not found: {file_path}")
except Exception as e:
      print(f"  [-] Error: Could not process file {file_path}. Error: {e}")

   return parsed_events


def main():      parser = argparse.ArgumentParser(description="A simple log correlation engine.")
parser.add_argument("--config", default="config.json", help="Path to the configuration file.")
parser.add_argument("log_files", nargs='*', help="Paths to specific log files to process (overrides
config paths).")

   args = parser.parse_args()

   try:        with open(args.config,
'r') as f:
         config = json.load(f)
except FileNotFoundError:
      print(f"[-]    Error:    Configuration    file    not    found    at
'{args.config}'")        return     except json.JSONDecodeError:
      print(f"[-] Error: Could not decode JSON from '{args.config}'")
return

   all_events = []
   log_sources = config.get('log_sources', [])

      # Use paths from config
for source in log_sources:
         all_events.extend(parse_log_file(source['path'], source))

   if not all_events:
      print("\n[-] No events were parsed. Exiting.")
return

   # Critical Step: Sort all events by timestamp to create a master timeline
all_events.sort(key=lambda x: x['timestamp_dt'])
   print(f"\n[+] Total events parsed and sorted: {len(all_events)}")

   incidents = correlate_events(all_events, config.get('correlation_rules', []))
```

```
    # --- Reporting ---
print("\n" + "="*60)
    print("                INCIDENT
REPORT")     print("="*60)     if not
incidents:
        print("No incidents found matching the defined correlation rules.")
else:
        for i, incident in enumerate(incidents, 1):
            print(f"\n--- Incident #{i} ---")
            print(f"Rule Matched: {incident['rule_name']}")
print(f"Correlated On: {incident['correlation_key']}")
            print(f"Time Window: {incident['start_time']} ->
{incident['end_time']}")            print("Events:")            for event_msg in
incident['events']:
                print(f"  - {event_msg}")
print("\n" + "="*60)

if __name__ == "__main__":
    main()
```

Output:

```
[+] Parsing auth.log...
[+] Parsing access.log...
  [!] Warning: Line 5 in access.log did not match regex.
[+] Total events parsed and sorted: 8

[+] Applying Rule: 'Failed SSH Brute-Force Attempt Followed by Success'
  [!] Incident Found: Failed SSH Brute-Force Attempt Followed by Success for 198.51.100.22

[+] Applying Rule: 'Admin Login Followed by Web Server Error'
  [!] Incident Found: Admin Login Followed by Web Server Error for 10.0.0.1
```

```
============================================================
                INCIDENT REPORT
============================================================

--- Incident #1 ---
Rule Matched: Failed SSH Brute-Force Attempt Followed by Success
Correlated On: ip_address: 198.51.100.22
Time Window: 2025-09-27 14:20:11 -> 2025-09-27 14:20:25
Events:
  - Sep 27 14:20:11 server sshd[1234]: Failed password for invalid user guest from 198.51.100.22 p
ort 12345 ssh2
  - Sep 27 14:20:15 server sshd[1235]: Failed password for root from 198.51.100.22 port 12346 ssh2
  - Sep 27 14:20:21 server sshd[1236]: Failed password for root from 198.51.100.22 port 12347 ssh2
  - Sep 27 14:20:25 server sshd[1237]: Accepted password for root from 198.51.100.22 port 12348 ss
h2

--- Incident #2 ---
Rule Matched: Admin Login Followed by Web Server Error
Correlated On: ip_address: 10.0.0.1
Time Window: 2025-09-27 15:25:10+0530 -> 2025-09-27 15:25:45+0530
Events:
  - 10.0.0.1 - admin [27/Sep/2025:15:25:10 +0530] "POST /admin/login HTTP/1.1" 200 147 "-" "Mozill
a/5.0"
  - 10.0.0.1 - admin [27/Sep/2025:15:25:45 +0530] "GET /internal/api/status HTTP/1.1" 500 58 "-"
"Mozilla/5.0"

============================================================
```