Task 02 : Debugging the Python Script

Bug 1: Incorrect Regular Expression Pattern

Problem Identified:

In the function **parse_log_line()** , the original regex pattern was too specific and incorrectly structured. It tried to extract "(\w+\.\w+\.\w+)" as the domain and "(\w+ /.+ HTTP/\d\.\d)" as the request.

On inspecting the regex pattern and comparing it with standard log formats, it was clear the pattern wouldn't match common cases like "GET /index.html HTTP/1.1"

The pattern expected \w+ /.+, which would fail if the method or path had unexpected characters.

Fix Applied:

Updated the regex to:
pattern = r'(\d+\.\d+\.\d+\.\d+) - (\w+) \[(\d{2}/\w{3}/\d{4}:\d{2}:\d{2}:\d{2} \+\d{4})\] "\S+" "\S+ .+ HTTP/\d\.\d" (\d{3}) (\d+) (\d+)'

Bug 2: Status Code Not Treated as Integer

Problem Identified:

In the function is_error_status() , the status code was being compared using numeric comparison (status >= 400), but it was returned as a string from match.groups().

When running the program received the following error: Exception has occurred: TypeError '>=' not supported between instances of 'str' and 'int'. Assuming status is an integer (like HTTP status codes 200, 404, etc.), but in reality, it's a string (e.g., '404' or '500').

Fix Applied:

Converting to int ensures code comparing numbers with numbers. The try … except block ensures code doesn't crash if status is not convertible to an integer.

```
def is_error_status(status):

    try:
     status = int(status)
     return 400 <= status <= 599
  except ValueError:
     return False
```

Bug 3: Final Window Not Checked

Problem Identified:

If the final time window in the log file doesn’t cross the 5-minute threshold, it is never evaluated for error rate.

When going through the code logic, after looping through all lines, the script doesn't re-check the final window’s error rate unless a new timestamp forces it.

Fix Applied:

Added a final check after the loop to evaluate the last batch of requests:

```
if window_requests > 0:
    error_rate = window_errors / window_requests
    if error_rate > error_threshold:
        print(f"Alert! Error rate {error_rate:.2%} exceeds threshold at {current_window_start}")
```

Bug 4: Error Rate Display Not in readable format

Problem Identified:

The error rate was printed as a float , which isn't easily interpreted by humans as a percentage.

print(f"Alert! Error rate {error_rate}% exceeds threshold at {current_window_start}")

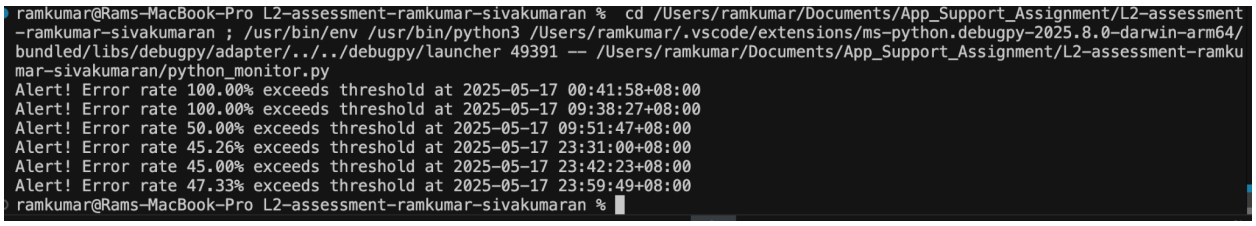
% after a float doesn't format it as a percentage—just appends the % symbol.

Fix Applied:

Used Python’s percentage formatting with two decimal places for clarity:

print(f"Alert! Error rate {error_rate:.2%} exceeds threshold at {current_window_start}")

Final output



Task 03 : Attack Pattern Detection Report

detect_attack_patterns.py Python script that parses the log file and detects potential attack patterns, including:

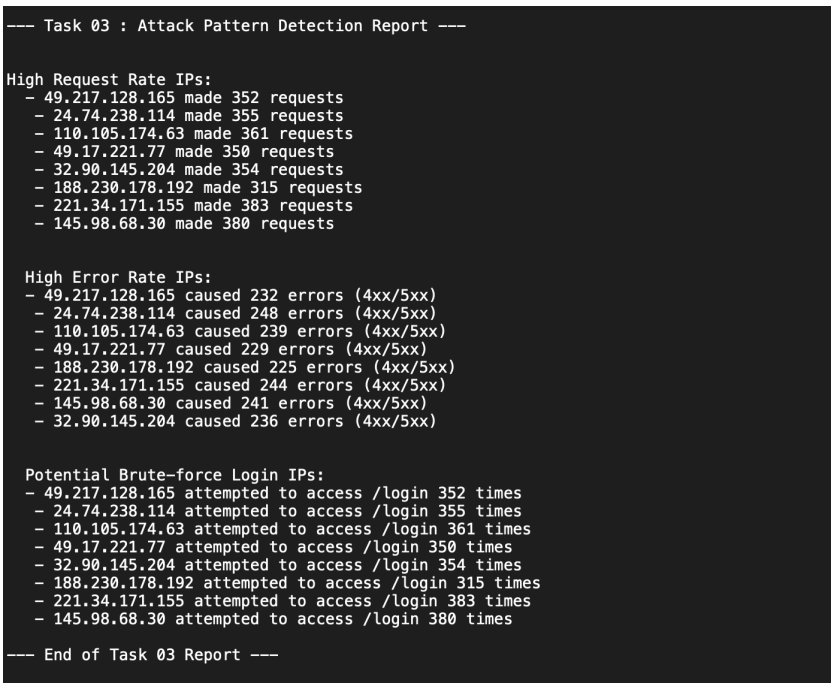
- High Request rate IPs.
- High Error Rate IPs.
- Potential Brute-force Login IPs.

Script mainly flows the following steps:

- Parses logs using regex
- Counts requests per IP
- Flags:
    - IPs with > threshold requests
      IPs with > threshold 4xx/5xx errors
    - IPs frequently hitting `/login`

Here for this assessment threshold values are assigned in a assumption as follows :

```
# Configuration thresholds
maxRequestsPerIp = 5
maxErrorsPerIp = 3
loginRequestThreshold = 3
```



```
--- Task 03 : Attack Pattern Detection Report ---


High Request Rate IPs:
  - 49.217.128.165 made 352 requests
   - 24.74.238.114 made 355 requests
   - 110.105.174.63 made 361 requests
   - 49.17.221.77 made 350 requests
   - 32.90.145.204 made 354 requests
   - 188.230.178.192 made 315 requests
   - 221.34.171.155 made 383 requests
   - 145.98.68.30 made 380 requests


  High Error Rate IPs:
  - 49.217.128.165 caused 232 errors (4xx/5xx)
   - 24.74.238.114 caused 248 errors (4xx/5xx)
   - 110.105.174.63 caused 239 errors (4xx/5xx)
   - 49.17.221.77 caused 229 errors (4xx/5xx)
   - 188.230.178.192 caused 225 errors (4xx/5xx)
   - 221.34.171.155 caused 244 errors (4xx/5xx)
   - 145.98.68.30 caused 241 errors (4xx/5xx)
   - 32.90.145.204 caused 236 errors (4xx/5xx)


  Potential Brute-force Login IPs:
  - 49.217.128.165 attempted to access /login 352 times
   - 24.74.238.114 attempted to access /login 355 times
   - 110.105.174.63 attempted to access /login 361 times
   - 49.17.221.77 attempted to access /login 350 times
   - 32.90.145.204 attempted to access /login 354 times
   - 188.230.178.192 attempted to access /login 315 times
   - 221.34.171.155 attempted to access /login 383 times
   - 145.98.68.30 attempted to access /login 380 times

--- End of Task 03 Report ---
```

Task 05 : CDN Performance Optimization

To enhance CDN and application layer performance, implementing **aggressive, behavior-aware traffic control and caching optimization at the CDN edge.**

1. Behavior-Based IP Rate Limiting

- **High Request Rate IPs**: Block IPs that exceed a predefined request threshold in a short time window.
- **High Error Rate IPs**: Identify IPs generating excessive 4xx/5xx errors and subject them to stricter rate limits or bans.
- **Brute-Force Login Attempts**: For IPs that repeatedly access /login endpoints, enforce multi-factor authentication after a certain number of attempts to mitigate brute-force login attacks.

2. Caching Optimization

- **Leverage CDN Caching**: Serve static assets (JavaScript, CSS, images) with extended TTL via Cache-Control headers to reduce origin traffic.
- **Conditional Requests**: Implement ETag and Last-Modified headers to support efficient client-side caching and reduce unnecessary data transfers.

3. Monitoring and Dynamic Adaptation

- Integrate real-time monitoring tools (Grafana, Prometheus, ELK stack) to:
    - Visualize traffic anomalies
    - Track effectiveness of rate-limiting and caching policies

---

Assessment done by: Ramkumar Sivakumaran

Github : https://github.com/Ramkumar96/L2-assessment-ramkumar-sivakumaran