Log Analyzer

Information Security Task2

Mohamed Ahmed Ramadan

2205043

Log Analysis

Log analysis provides automated processing and insights from web server access logs. It transforms raw data into actionable metrics for performance monitoring, error debugging, and security auditing

Key Features:

1. **Quantify Traffic Patterns**
    - Calculate total requests, methods (GET/POST), and unique IPs
    - Identify peak activity hours and daily averages
2. **Monitor System Health**
    - Detect failed requests and compute failure rates
    - Pinpoint high-error time periods
3. **Support Security & Optimization**
    - Flag most active users and suspicious IPs
    - Generate actionable insights for capacity planning and debugging

Steps for creating log analyzer:

1. **Data collection :** collect comprehensive log data
2. **Script development:** creating bash script that is able to handle the data and analyze it
3. **Feedback analysis**: give security suggestions based on finding

1. Data collection

Data name & source *: Web Server Access Logs* , *Harvard online shopping web logs*

Data size : 3.52 GB (10,365,109 rows x 12 col)

About this data: Web sever logs contain information on any event that was registered/logged. This contains a lot of insights on website visitors, behavior, crawlers accessing the site, business insights, security issues, and more. logs from an Iranian ecommerce website zanbil.ir.

Column Descriptions

1. **IP**: The client's IP address making the request.
2. **Timestamp**: Date and time of the request in %d/%b/%Y:%H:%M:%S format .
3. **Method :** HTTP request method (e.g., GET, POST, HEAD).
4. **URL :**The requested resource path (e.g., /index.html).
5. **Protocol** :HTTP protocol version (e.g., HTTP/1.1).
6. **status code**: HTTP response status code (e.g., 200 for success, 404 for not found).
7. **bytes sent**: Size of the response in bytes (0 if missing).
8. **user agent**: Client's browser/device identifier (e.g., Mozilla/5.0).
9. **url_length**: Character count of the URL (security/analysis metric).
10. **url_depth**: Number of / in the URL (indicates nested paths).
11. **num_encoded_chars**: Count of URL-encoded characters .
12. **num_special_chars** : Count of special characters (|, ,, ;) in URLs (security flag).

note : I have processed data so its easier to read and deal with . by structuring the data , parsing & cleaning the data . the new data is uploaded in the repository

2. Log Analyzer Bash Script

1. **Initialization & Setup**

- Check for the existence of input file ensuring error handling
- Creating output file to report all the findings

```bash
#!/bin/bash

# Check if input file exists
if [ ! -f "cleaned_access_log.csv" ]; then
    echo "Error: cleaned_access_log.csv not found!" > log_analysis_report.txt
    exit 1
fi

# Initialize report file
report_file="log_analysis_report.txt"
{
echo "Log File Analysis Report"
echo "Generated on: $(date +"%Y-%m-%d %H:%M:%S")"
echo "Data Source: cleaned_access_log.csv"
echo ""
```

2. **Request Analysis**

- Count total requests
- Count GET total requests
- Count POST total requests

```bash
# 1. Request Counts
echo "=== BASIC STATISTICS ==="
total_requests=$(wc -l < cleaned_access_log.csv)
echo "Total Requests: $total_requests"

get_requests=$(awk -F',' '$3 == "GET" {count++} END {print count}'
cleaned_access_log.csv)
post_requests=$(awk -F',' '$3 == "POST" {count++} END {print count}'
cleaned_access_log.csv)
echo "GET Requests: $get_requests"
echo "POST Requests: $post_requests"
```

3. **Unique IP count**

- Extract total IP count and remove duplicate to find the total count of distinct IPs that accessed the server

```
# 2. Unique IP Addresses
unique_ips=$(cut -d',' -f1 cleaned_access_log.csv | sort -u | wc -l)
echo "Unique IP Addresses: $unique_ips"
echo ""
```

4. **Failure Analysis**

- Filters rows where status code is between 400-500(error status) , Counts matching lines.
- Computes percentage of failed requests vs total requests

```
# 3. Failure Requests
echo "=== FAILURE ANALYSIS ==="
failed_requests=$(awk -F',' '$6 ≥ 400 && $6 ≤ 500 {count++} END {print count}
' cleaned_access_log.csv)
failure_percentage=$(awk -v failed=$failed_requests -v total=$total_requests
'BEGIN {printf "%.2f", (failed/total)*100}')
echo "Failed Requests: $failed_requests"
echo "Failure Percentage: $failure_percentage%"
echo ""
```

5. **Top user analysis**

- Counts all IPs, sorts by frequency, shows top entry
- Identifies the overall , GET & POST most frequent visitor by IP

```
# 4. Top Users
echo "=== TOP USERS ==="
echo "Most Active IP Overall:"
cut -d',' -f1 cleaned_access_log.csv | sort | uniq -c | sort -nr | head -1
echo ""

echo "Top GET Requester:"
awk -F',' '$3 == "GET" {print $1}' cleaned_access_log.csv | sort | uniq -c |
sort -nr | head -1
echo ""

echo "Top POST Requester:"
awk -F',' '$3 == "POST" {print $1}' cleaned_access_log.csv | sort | uniq -c |
sort -nr | head -1
```

6. **Temporal Analysis**

- Calculates average daily traffic by Dividing total requests by days
- Shows worst 3 days for errors by Counting errors per day.
- Top 5 busiest timestamps by Counting requests per full timestamp
- Timestamps with most errors by Filtering errors before counting

```
# Daily Averages
total_days=$(awk -F',' '{print substr($2,2,11)}' cleaned_access_log.csv | sort
-u | wc -l)
avg_daily_requests=$(awk -v total=$total_requests -v days=$total_days 'BEGIN
{printf "%.2f", total/days}')
echo "Average Daily Requests: $avg_daily_requests"
echo ""

# Failure Days (Top 3)
echo "=== TOP FAILURE DAYS ==="
awk -F',' '$6 ≥ 400 && $6 ≤ 500 {print substr($2,2,11)}' cleaned_access_log.
csv | sort | uniq -c | sort -nr | head -3
echo ""

# Peak Traffic Times (Top 5)
echo "=== PEAK TRAFFIC TIMES ==="
awk -F',' '{print $2}' cleaned_access_log.csv | sort | uniq -c | sort -nr |
head -5
echo ""

# Failed Request Times (Top 5)
echo "=== FAILURE PEAK TIMES ==="
awk -F',' '$6 ≥ 400 && $6 ≤ 500 {print $2}' cleaned_access_log.csv | sort |
uniq -c | sort -nr | head -5
echo ""
```

7. **Status code breakdown**

- Groups codes between 200-500 , show Frequency of all valid status codes

```
# 6. Status Code Breakdown
echo "=== STATUS CODE BREAKDOWN ==="
awk -F',' '$6 ≥ 200 && $6 ≤ 500 {count[$6]++} END {for (code in count) print
count[code], code}' cleaned_access_log.csv | sort -nr
echo ""
```

3. Output & Suggestion

Script analysis:

```
=== BASIC STATISTICS ===
Total Requests: 10365110
GET Requests: 10190005
POST Requests: 139155
Unique IP Addresses: 258601

=== FAILURE ANALYSIS ===
Failed Requests: 160573
Failure Percentage: 1.55%

=== TOP USERS ===
Most Active IP Overall:
 353483 66.249.66.194

Top GET Requester:
 353483 66.249.66.194

Top POST Requester:
  11712 151.239.241.163

=== TEMPORAL ANALYSIS ===
Average Daily Requests: 647819.38

=== TOP FAILURE DAYS ===
  22777 019-01-26 1
  20473 019-01-23 1
  19984 019-01-22 1

=== PEAK TRAFFIC TIMES ===
    368 2019-01-26 19:07:39
    323 2019-01-26 19:05:58
    313 2019-01-26 16:06:25
    309 2019-01-26 12:39:19
    298 2019-01-26 09:59:47

=== FAILURE PEAK TIMES ===
     17 2019-01-26 14:11:28
     17 2019-01-25 12:00:37
     17 2019-01-22 16:19:25
     16 2019-01-25 19:43:11
     16 2019-01-24 10:58:09
```

=== STATUS CODE BREAKDOWN ===
9100330 200
340228 304
182636 302
104957 404
63376 301
47460 499
5492 403
1736 500
323 401
321 400
109 408
6 405
3 206

=== SUGGESTIONS ===
1. Immediate Actions:
  - Investigate failures on 22777 019-01-26 1
  - Check server logs during peak failure time: 17 2019-01-26 14:11:28

2. Security Review:
  - Monitor GET requests from IP: 353483 66.249.66.194
  - Monitor POST requests from IP: 11712 151.239.241.163

3. Performance Optimization:
  - Scale resources during peak traffic times shown above
  - Review endpoints returning 4xx/5xx errors in status breakdown