Name: Ahmed Mahmoud Hassan
ID: 2205155

Introduction

This report presents a comprehensive analysis of a server log containing 500 HTTP requests. Using a custom Bash script, we examine the log to extract critical security-related metrics, identify patterns of access, and highlight potentially malicious behaviors. The goal of this analysis is to provide actionable insights for improving the security and performance of the web server.

First I got my sample to run the script and extract the log from containing 500 Requests

log_analysis > sample.log

```
192.168.1.26 - - [08/Feb/2023:12:24:12 +0000] "GET /contact HTTP/1.1" 404 1234
192.168.1.24 - - [09/Feb/2023:00:03:54 +0000] "POST /search HTTP/1.1" 200 56
192.168.1.11 - - [13/Feb/2023:18:52:17 +0000] "GET /profile HTTP/1.1" 200 1024
192.168.1.23 - - [08/Feb/2023:03:39:17 +0000] "POST /search HTTP/1.1" 200 56
192.168.1.28 - - [11/Feb/2023:09:56:09 +0000] "GET /search HTTP/1.1" 200 1024
192.168.1.46 - - [07/Feb/2023:05:56:16 +0000] "POST /login HTTP/1.1" 500 1024
192.168.1.16 - - [11/Feb/2023:15:02:31 +0000] "GET /about.html HTTP/1.1" 200 1234
192.168.1.13 - - [09/Feb/2023:02:15:47 +0000] "GET /login HTTP/1.1" 403 56
192.168.1.27 - - [12/Feb/2023:17:37:38 +0000] "GET /login HTTP/1.1" 500 1234
192.168.1.10 - - [12/Feb/2023:00:52:46 +0000] "GET /index.html HTTP/1.1" 403 3421
192.168.1.33 - - [09/Feb/2023:08:21:22 +0000] "GET /contact HTTP/1.1" 404 1234
192.168.1.26 - - [05/Feb/2023:08:34:43 +0000] "GET /search HTTP/1.1" 200 3421
192.168.1.34 - - [10/Feb/2023:12:55:01 +0000] "POST /search HTTP/1.1" 403 1024
192.168.1.14 - - [08/Feb/2023:11:20:26 +0000] "POST /index.html HTTP/1.1" 200 3421
192.168.1.21 - - [11/Feb/2023:21:13:25 +0000] "POST /contact HTTP/1.1" 200 3421
192.168.1.45 - - [12/Feb/2023:19:58:31 +0000] "POST /search HTTP/1.1" 200 1234
192.168.1.30 - - [11/Feb/2023:03:53:32 +0000] "POST /api/data HTTP/1.1" 403 1024
192.168.1.45 - - [06/Feb/2023:14:49:26 +0000] "GET /about.html HTTP/1.1" 403 56
192.168.1.33 - - [06/Feb/2023:07:42:09 +0000] "GET /index.html HTTP/1.1" 200 789
192.168.1.1 - - [13/Feb/2023:10:49:07 +0000] "GET /profile HTTP/1.1" 403 789
192.168.1.21 - - [09/Feb/2023:16:08:14 +0000] "GET /api/data HTTP/1.1" 200 56
```

Then I have created my bash script file

```
$ analyze_log.sh
```

Script Design and Methodology:

The Bash script was developed to automate the parsing and analysis of the HTTP access log. The script executes the following major functions:

1. **Log File Validation**
    - **Ensures the log file exists and is readable before proceeding.**
2. **Request Counting**
    - **Total number of requests.**
    - **Breakdown of HTTP methods (GET and POST).**
3. **IP Address Analysis**
    - **Counts unique IP addresses.**
    - **Identifies request frequency per IP.**
4. **Failure Detection**
    - **Calculates failed requests and computes failure rate percentages.**
5. **Top Active IPs**
    - **Identifies IPs with the highest request count.**
6. **Temporal Analysis**
    - **Observes patterns in traffic across days and hours.**
7. **HTTP Status and Method Breakdown**
    - **Provides insights into server behavior and usage trends.**

Let me break it down for you:

```bash
#!/bin/bash

if [ $# -eq 0 ]; then
    echo "Usage: $0 <log_file>"
    exit 1
fi

LOG_FILE=$1
REPORT_FILE="log_analysis_report_$(date +%Y%m%d).txt"

if [ ! -f "$LOG_FILE" ]; then
    echo "Error: File $LOG_FILE not found!"
    exit 1
fi
```

Here we check if the log file was created and check if it exists

```bash
total_requests=$(wc -l < "$LOG_FILE")
get_requests=$(grep -c 'GET' "$LOG_FILE")
post_requests=$(grep -c 'POST' "$LOG_FILE")
```

Here are the basic requests counting which counts for total requests (wc -l) and counts the get and post requests by each (grep -c 'POST/GET')

```bash
unique_ips=$(awk '{print $1}' "$LOG_FILE" | sort -u | wc -l)
ip_method_counts=$(awk '{print $1,$6}' "$LOG_FILE" | sort | uniq -c | sort -nr)
```

Here it analyze the unique ip address it first extract the ip addresses and then removes the duplicates for unique count and counts the requests per IP

```bash
failed_requests=$(awk '$9 ~ /^[45][0-9][0-9]$/' "$LOG_FILE" | wc -l)
failure_percentage=$(awk "BEGIN {printf \"%.2f\", ($failed_requests/$total_requests)*100}")
```

Identifying the failed requests and calculate it with percentage

```bash
top_ip=$(awk '{print $1}' "$LOG_FILE" | sort | uniq -c | sort -nr | head -n 1)
```

Finding the most active IP address

```
daily_requests=$(awk -F'[' '{print $2}' "$LOG_FILE" | awk '{print $1}' | sort | uniq -c)
total_days=$(echo "$daily_requests" | wc -l)
avg_daily=$(awk "BEGIN {printf \"%.2f\", $total_requests/$total_days}")

failure_days=$(awk '$9 ~ /^[45][0-9][0-9]$/' "$LOG_FILE" | awk -F'[' '{print $2}' | awk '{print $1}' | sort | uniq -c | sort -nr)


hourly_requests=$(awk -F'[:[]' '{print $3}' "$LOG_FILE" | awk -F: '{print $1}' | sort | uniq -c)
```

Analyzing temporal patterns in requests

```
status_codes=$(awk '{print $9}' "$LOG_FILE" | sort | uniq -c | sort -nr)

top_get_user=$(awk '$6 == "\"GET" {print $1}' "$LOG_FILE" | sort | uniq -c | sort -nr | head -n 1)
top_post_user=$(awk '$6 == "\"POST" {print $1}' "$LOG_FILE" | sort | uniq -c | sort -nr | head -n 1)

failure_hours=$(awk '$9 ~ /^[45][0-9][0-9]$/' "$LOG_FILE" | awk -F'[:[]' '{print $3}' | awk -F: '{print $1}'
```

Status code and method analysis it provides detailed HTTP and behavioral analysis

```
{
    echo "=== LOG ANALYSIS REPORT ==="
    echo "Generated on: $(date)"
    echo "Analyzed file: $LOG_FILE"
    echo ""

    echo "1. REQUEST COUNTS"
    echo "   Total requests: $total_requests"
    echo "   GET requests: $get_requests"
    echo "   POST requests: $post_requests"
    echo ""

    echo "2. UNIQUE IP ADDRESSES"
    echo "   Total unique IPs: $unique_ips"
    echo "   Requests per IP and method:"
    echo "$ip_method_counts"
    echo ""
```

Script Execution and Output Formatting
Upon execution, the script processes the log file and prints a structured, human-readable report. Key data points are extracted using tools like grep, awk, sort, and uniq, with sections clearly labeled for interpretability. Below is a description of each output metric.
Then here we start to generate and print our script while formatting the data (500 requests) into a structured report

We start by running the command→

```
$ ./analyze_log.sh sample.log
```

To create the LOG FILE

log_analysis_report_20250510.txt

```
1. REQUEST COUNTS
   Total requests: 500
   GET requests: 259
   POST requests: 241
```

The request counter counts all the requests and tells us how many POSTs & GETs methods are there

```
2. UNIQUE IP ADDRESSES
   Total unique IPs: 50
   Requests per IP and method:
     12 192.168.1.34 "POST
     11 192.168.1.8 "POST
     11 192.168.1.41 "GET
     11 192.168.1.26 "GET
     11 192.168.1.16 "GET
     10 192.168.1.35 "GET
      9 192.168.1.49 "POST
      9 192.168.1.32 "POST
      9 192.168.1.29 "POST
```

And here shows all the unique IP addresses that made requests

```
3. FAILURE REQUESTS
   Failed requests (4xx/5xx): 240
   Failure percentage: 48.00%
```

Here shows the failure requests with a 48%

```
4. TOP USER
   Most active IP:      19 192.168.1.41
```

Here shows the most user that made requests (POSTs/GETs)

```
5. DAILY REQUEST AVERAGES
   Average requests per day: 1.00
   Daily request counts:
      1 05/Feb/2023:00:02:39
      1 05/Feb/2023:00:02:41
      1 05/Feb/2023:00:04:36
      1 05/Feb/2023:00:38:01
      1 05/Feb/2023:01:29:54
      1 05/Feb/2023:01:53:11
```

This shows the daily requests average for each day

```
6. FAILURE ANALYSIS
   Days with most failures:
      1 14/Feb/2023:22:44:09
      1 14/Feb/2023:22:25:16
      1 14/Feb/2023:22:08:27
      1 14/Feb/2023:20:53:03
      1 14/Feb/2023:17:09:21
      1 14/Feb/2023:16:22:15
```

And the failure analysis in what time and day a request has failed

```
ADDITIONAL METRICS
a. Hourly request distribution:
    24 00
    20 01
    17 02
    16 03
    18 04
```

```
b. Status code breakdown:
   260 200
    88 403
    77 500
    75 404

c. Most active users by method:
   Top GET user:      11 192.168.1.41
   Top POST user:      12 192.168.1.34

d. Failure patterns by hour:
    16 15
    14 17
    14 06
```

And here shows the additional metrice showing the hourly request distribution, status code breakdown, most active users in POST / GET and the failure pattern by hour

```
ANALYSIS SUGGESTIONS
1. Failure reduction:
   - Investigate top error codes: 403,500,404,
   - Focus on peak failure hours: 15:00 (16 failures),17:00 (14 failures),06:00 (14 failures),

2. Performance optimization:
   - Highest traffic hour: 16:00 (29 requests)
   - Consider load balancing or caching during peak times

3. Security recommendations:
   - Review activity from top POST user (     12 192.168.1.34)
   - Check for brute force patterns (many rapid failures from single IP)
   - Monitor suspicious user agents or unusual request patterns
```

Here we got our suggestions on how to fix these errors and while maintaining security too

- **We should investigate the repeated server errors (403,500,404)**
- **And focus on failure hours which we can see that at 15:00 o'clock we have 16 failures and for 17:00 o'clock we have 14 and so on**
- **We should maintain high traffic too we can see that at 16:00 o'clock we got 29 requests we should handle their requests with no server breakdown or any conflicts**
- **And we should consider load balancing and caching during this time**
- **We have to review the POST reviews from the user (192.168.1.34) since he have the most POST requests so we check if he didn't try to inject any malicious code**
- **And checking the brute force patterns since there were too many failures from same IP in a short period of time**
- **And we should monitor the users requests to eject any suspicious request before it breaks the server or inject a malicious code into it which can lead into data leakage**

Conclusion

This report demonstrates how automated log analysis can uncover valuable security insights. The findings highlight the importance of continuous monitoring, user behavior analysis, and timely mitigation to protect web infrastructure from evolving threats.