LOG FILE ANALYSIS REPORT

Student Name: Jerome Arsany Mansour Farah

Id: 2305093 (Level 2)

Course: Information Security Mangement

1. INTRODUCTION

This document presents an in-depth analysis of a web server access log file consisting of **10,000 HTTP requests**, which were collected over a four-day period from **17 May 2015 to 20 May 2015**. The log file was sourced from a sample dataset made publicly available by Elastic, commonly used for practicing log analytics and cybersecurity auditing. It reflects real-world web server traffic, including various request methods, status codes, and access patterns from a broad range of IP addresses.

The log was processed using a custom Bash script executed in a Kali Linux environment. The objective of this analysis is to extract meaningful usage patterns, detect anomalies in traffic, assess HTTP method distribution, evaluate response codes, and examine temporal trends such as failures per hour or day. Particular attention is given to potential security concerns such as scanning behavior, bot activity, and system-level errors. Based on the findings, the report concludes with a set of recommendations aimed at improving server security posture, monitoring practices, and log management procedures.

2. DATASET SUMMARY

The provided data includes:

1. Total number of HTTP requests: 10,000

2. Request types and method counts

3. List of most active IPs and how many requests they made

4. Breakdown of HTTP response status codes

5. Number of unique IPs

6. Failure statistics by day and hour

7. Time span of the logs

8. The data appears to cover four consecutive days: May 17 to May 20, 2015.

It is a high-level summary derived from a log parser script, not the raw log entries. As a result, analysis of finer-grained patterns (e.g., specific URLs, user agents, referers, or request payloads) is not possible here, but the available data still reveals significant insights.

3. TRAFFIC VOLUME ANALYSIS

Total number of requests: 10,000

From the reported timeframe **(May 17–20)**, this means an average of about **2,500 requests per day**. When distributed across 24 hours, this equates to around **104 requests per hour**, assuming evenly distributed traffic. However, further time-based breakdowns reveal that traffic was not completely uniform and some hours had more activity than others.

This volume represents moderate web traffic. For a small to medium-sized site, this is typical and sustainable. However, depending on the server's resources, even moderate levels of malicious or non-human traffic could begin to degrade performance or cause issues if left unmanaged.

4. HTTP METHOD ANALYSIS

GET requests: 9,952

POST requests: 5

The GET method is overwhelmingly dominant in this log file, which is typical for static or read-oriented websites. GET requests are used to retrieve data, such as HTML pages, images, stylesheets, etc.

POST requests are used to send data to the server, usually when submitting forms or interacting with APIs. The very small number of POST requests (only 5 across four days) suggests one of the following:

1. The site has very few interactive features or users.
2. The logging script excluded or filtered out non-GET traffic.
3. Bots and crawlers are dominating the traffic, and bots rarely submit forms.

This implies a low risk of direct injection attacks via POST, but the overall GET dominance raises concerns about automated scraping or reconnaissance.

5. UNIQUE IP ADDRESS ANALYSIS

Number of unique IP addresses: 1,753

This indicates the number of distinct hosts (clients) accessing the server. With 10,000 requests, this means each IP made, on average, about **5–6 requests**. However, this average is misleading because of heavy activity from a few specific IP addresses.

A deeper look into the top IPs shows highly skewed request distribution. A small number of IPs account for a large percentage of the total traffic.

This pattern often appears when:

- Search engines are crawling the site (e.g., Googlebot, Bingbot)
- Malicious actors are scanning the server
- Misconfigured scripts are causing excessive requests

---

6. MOST ACTIVE IP ADDRESSES

The most active IP was **"66.249.73.135"**, which made 482 GET requests. This IP is associated with Googlebot, a legitimate web crawler used by Google to index sites.

Other high-activity IPs include:

- **"46.105.14.53"** – made 364 requests
- **"130.237.218.86"** – made 357 requests
- **"75.97.9.59"** – made 273 requests

In total, more than 50 IPs made between 30 and 100+ requests, which suggests regular polling or automated scraping.

These patterns raise questions:

1. Are these bots respecting the site's robots.txt file?
2. Are any of these IPs not associated with known services and may require throttling or blocking?

IPs like **"66.249.73.135"** should be allowed, but other IPs should be reviewed using WHOIS, reverse DNS, or threat intelligence tools to determine if they pose a risk.

---

7. HTTP STATUS CODE BREAKDOWN

Most common status codes:

200 OK: 9126

304 Not Modified: 445

404 Not Found: 213

301 Moved Permanently: 164

206 Partial Content: 45

500 Internal Server Error: 3

403 Forbidden: 2

416 Range Not Satisfiable: 2

Interpretation:

200 OK responses are expected and indicate successful content delivery.

304 Not Modified suggests content is being cached appropriately, which is good for performance.

404 Not Found responses are common but should be reviewed. If attackers are scanning for vulnerabilities, they may request random or malicious paths.

301 Moved Permanently responses suggest some URLs have been redirected. High numbers may slow down page load times and confuse bots if the redirects are unnecessary.

206 Partial Content is seen in media or file downloads. 45 such responses suggest some download or media activity.

500 Internal Server Error is a critical issue. Even just 3 such errors may indicate broken server-side logic, resource exhaustion, or misconfigured scripts.

403 Forbidden means access was intentionally blocked, a good thing if it's protecting sensitive resources.

416 Range Not Satisfiable is rare and may indicate malformed requests, often used in vulnerability testing tools or malformed scanners.

---

8. FAILURE ANALYSIS BY DAY

Days with the most failed requests:

May 18: 66 failures

May 19: 66 failures

May 20: 58 failures

May 17: 30 failures

A clear spike in failed requests is visible on May 18 and 19. These dates should be reviewed more carefully in the original log files to see if there were concentrated scans, attacks, or server issues.

Failure spikes may indicate:

- A new vulnerability being tested by bots
- A misconfiguration or bug introduced on the server
- An external attack attempting to gain access

---

9. FAILURE ANALYSIS BY HOUR

Hours with the most failures:

09:00: 18 failures

05:00: 15 failures

06:00: 14 failures

These early morning hours are outside normal usage patterns for most human users. Automated tools, such as scanners or scrapers, often operate during low-traffic hours to avoid detection.

A pattern of failures during off-hours may indicate:

1. Coordinated vulnerability scanning
2. Resource testing (range attacks, path enumeration)
3. Bots running via cron jobs or scheduled scripts

---

10. SECURITY RISKS AND OBSERVATIONS

1. The small number of POST requests may reduce input-based attack risk, but GET requests can also carry malicious payloads in URLs.

2. Multiple requests from the same unknown IP addresses raise concerns about scraping, spam, or brute force attempts.

3. Error spikes at certain times strongly suggest automated activity.

4. HTTP status 500, although rare, is dangerous. Even one internal server error could allow attackers to learn about backend technology, configurations, or flaws.

---

11. RECOMMENDATIONS FOR IMPROVEMENT

1. Rate Limiting: Use fail2ban, mod_evasive, or other tools to block IPs making too many requests in a short time.

2. Bot Filtering: Identify and allow only verified bots like Googlebot; block unknown ones based on User-Agent or IP reputation.

3. Security Logging: Always log full details including User-Agent, Referer, full URLs, and timestamps. These are essential for deep analysis.

4. Error Handling: Review the source of status 500 errors. Fix broken scripts or resource limits causing these.

5. Monitoring: Set up automated alerting if failure rates exceed a certain threshold within a short time period.

6. File Integrity Monitoring: If suspicious behavior is observed, check server files for unauthorized changes.

7. DNS and WHOIS Lookups: Periodically analyze high-traffic IPs to validate legitimacy.

8. Update Web Application Firewall Rules: Tailor WAF rules to your traffic profile to block malicious patterns more effectively.

12. CONCLUSION

The log data reveals a moderately trafficked web server with a heavy skew toward automated requests and search engine crawlers. The small number of POST requests and the dominance of GET activity suggest low interactivity, but this also simplifies the monitoring process.

The most notable risks are:

1. Frequent automated access
2. Repeated failures at certain times
3. A small set of IPs dominating the traffic

By tightening request handling, expanding log detail, and introducing smarter monitoring and blocking tools, the server's security and reliability can be significantly improved.