# **Introduction**

Web traffic data comprising a series of logs spanning 30 days was analysed to identify indicators of unusual or potentially malicious activity targeting a simulated Apache web server environment.

The analysis identified 40 security alerts across three categories:

* High-frequency IP traffic, which suggests automated scanning
* Failed authentication bursts against sensitive endpoints
* Repeated probing of high-risk administrative paths

While there weren’t any successful compromises in the dataset, the activities indicate ongoing reconnaissance and brute-force attempts that could pose a serious risk if left unmonitored.

# **Scope & Method**

## Scope

The analysis focused on incoming HTTP requests captured in 30 days of Apache access logs. Its main objective was to detect patterns of suspicious activity to alert about dangers within the network. These dangers highlight areas of compromise that recommendations aim to mitigate. The idea was to prevent such issues from recurring.

## Method

A Python based parsing script called parse.py got developed for the method. It extracted key fields from the logs. Examples of the fields are IP address, timestamp, status code, and path. The script generated summary statistics drawn from the traffic data in the logs. It applied specific detection rules too. Those rules identified suspicious activity and triggered alerts based on the log contents. Findings were then exported into structured reports for review.

# **Data Sources**

## Files

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Description** | **Format** | **Volume** |
| sample\_30days.log | Apache access logs over period of 30 days | Plain text log file | 6180 entries of logs |
| alerts.csv | Suspicious activities detected based on the rules implemented | CSV file | 41 rows & 5 columns |
| summary.csv | Parsed traffic summary of the logs | CSV file | 37 rows & 5 columns |

## Key Log Fields:

1. **IP Address** – identifies the device within the network
2. **Timestamp** – tells the exact moment an event happened
3. **HTTPS** **Status** **Code** – indicates whether an HTTP request has been completed
4. **Request** **Path** – shows the resource the user is trying to access
5. **User** **Agent** – identifies what software is making the request
6. **Event** **Type** – defines what type of activity is being conducted – in **alerts.csv**
7. **Record** **Type** – what type of log record – in **summary.csv**

# **Detection Logic & Rules**

## Rules implemented within ‘parse.py’

|  |  |  |  |
| --- | --- | --- | --- |
| **Rule Name** | **Condition** | **Severity** | **Purpose** |
| HighFrequency | >= 200 requests from a single IP address | Medium/High | Detect very high traffic |
| FailedLoginBurst | >= 5 failed authentication attempts, with a 401/403 error to ‘/api/auth’ | High | Detect brute-force login attempts |
| SensitivePathProbe | >= 3 attempts on sensitive paths | High | Detect probing for known misconfigurations and admin panels |

**401 error** is when a user within a network isn’t recognised, while a **403 error** recognises the user but they have unauthorised access to the service (Manor, 2023).

## Sensitive paths monitored:

1. /admin – admin panel login page
2. /wp-admin – WordPress admin login dashboard
3. /phpmyAdmin – phpMyAdmin database management interface (docs.phpmyadmin.net, n.d.)
4. /server-status – status of the server
5. /.env – simple text file holding key-value pairs (Ellen, 2024)
6. /.git/ - directory within a Git project
7. /api/auth – endpoint for API authorisation, which refers to allowing or denying access to an API (Akamai, 2025)

These paths are sensitive because they are critical access, sensitive data and authentication points. Any unauthorised access to any of them can give attackers the sensitive credentials & data they need, to launch malicious activities on it or keep the data for themselves.

# **Key Findings**

|  |  |
| --- | --- |
| 18 IP addresses exceeded the “HighFrequency” threshold, with the highest at 385 requests. |  |
| 4 IP addresses triggered the “FailedLoginBurst” rule against “/api/auth”. |  |
| 18 IP addresses repeatedly accessed sensitive endpoints, which breached the "SensitivePathProbe" rule, such as `/admin` and `/.env` |  |
| High amounts of 403 (Forbidden) and 500 (Server Error) status codes, which indicate multiple failed or blocked access attempts. |  |

The screenshots of the first three findings were from the “alerts.csv” file, while the final finding was from the “summary.csv” file, which had 2 screenshots. One shows the log fields, and the other shows only the part of the CSV file that matches the finding itself.

# **Activity Breakdown**

All breakdowns were taken from “summary.csv”

## Top 10 IP addresses by Volume

1. 192.168.1.13 - 385 times
2. 192.168.1.9 - 376 times
3. 192.168.1.11 - 366 times
4. 203.0.113.9 - 362 times
5. 192.168.1.14 - 359 times
6. 203.0.113.8 - 355 times
7. 192.168.1.6 - 343 times
8. 192.168.1.10 - 343 times
9. 203.0.113.6 - 342 times
10. 192.168.1.2 - 342 times

## HTTP Status Code Counts

1. 200 – 3084 times
2. 403 – 789 times
3. 404 – 759 times
4. 500 – 790 times
5. 302 – 758 times

## Access of Sensitive Path Counts

1. /products – 516 times
2. /admin – 505 times
3. /search – 502 times
4. /.env – 493 times
5. /login – 483 times
6. /,475 times
7. /api/data – 470 times
8. /images/logo.png – 469 times
9. /.git/config – 466 times
10. /phpMyAdmin – 464 times
11. /dashboard – 448 times
12. /wp-admin – 447 times
13. /api/auth – 442 times

# **Alerts Summary**

|  |  |  |
| --- | --- | --- |
| **Event Type** | **Alerts** | **Description** |
| HighFrequency | 18 | Excessive traffic volumes, indicating scanning behaviour |
| FailedLoginBurst | 4 | Brute-force attacks resulting in multiple authentication failures |
| SensitivePathProbe | 18 | Access to known sensitive endpoints |

## Notable IP addresses

* 192.168.1.13 – 385 requests (HighFrequency) and 162 sensitive hits (SensitivePathProbe).
* 192.168.1.2 – 6 failed authentications (FailedLoginBurst) and 128 sensitive hits (SensitivePathProbe).
* 203.0.113.7 - 333 requests (HighFrequency), 125 sensitive hits (SensitivePathProbe) and 5 failed authentications (FailedLoginBurst).
* 203.0.113.9 - 362 requests (HighFrequency), 129 sensitive hits (SensitivePathProbe) and 6 failed authentications (FailedLoginBurst).
* 192.168.1.12 – 300 requests (HighFrequency) and 128 sensitive hits (SensitivePathProbe).

# **Risk Assessment**

The table below shows the risk and potential malicious behaviours, which will need to be mitigated and monitored.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Threat Actor** | **Likelihood** | **Impact** | **Risk Level** | **Comment** |
| Brute-force attacks | High | Medium | High | Multiple IP addresses repeatedly target ‘/api/auth’ |
| Sensitive path access attempts | High | High | High | Evidence of looking for weaknesses within system |
| High-frequency scanning | High | High | High | Results in high traffic within the network |

# **Recommendations**

## Rate Limiting & Blocking

You can start by handling rate limiting and blocking. Set up request limits based on IP addresses. Add temporary bans when someone makes too many in a short time. This approach really cuts down on all that scanning.

## Authentication

Next, think about hardening authentication. Enable multi-factor authentication for the critical endpoints that matter most. Also, implement account lockouts and CAPTCHA checks after repeated failed login attempts. It makes things a lot tougher for attackers.

## Restrict Sensitive Paths

When it comes to restricting sensitive paths, you have options. Move access to spots like /admin, /.env, or /phpmyadmin behind allow-lists. On top of that, remove any exposed .git directories. Secure those environment files so they stay hidden.

## Improve Error Handling

Error handling needs some work, too. Go through and review what causes those 500 errors, which often reveal vulnerabilities.

## SIEM Integration

Finally, integrate with SIEM tools for better oversight by automating log parsing and setting up alerts using tools like Splunk. SIEM tools provide real-time monitoring and the ability to set up dashboards to track everything easily.

# **References**

1. Manor, G.L. (2023). *401 vs. 403 Error Codes: What’s the Difference? When to Use Each? (Updated 2024)*. [online] Permit.io. Available at: <https://www.permit.io/blog/401-vs-403-error-whats-the-difference> [Accessed 09 Oct. 2025].
2. docs.phpmyadmin.net. (n.d.). *Introduction — phpMyAdmin 5.1.0-dev documentation*. [online] Available at: <https://docs.phpmyadmin.net/en/latest/intro.html> [Accessed 09 Oct. 2025].
3. Ellen, L. (2024). *What is .env files in Python? Managing Environment Variable & Securing Your Secrets*. [online] Medium. Available at: <https://python.plainenglish.io/the-essentials-of-env-files-in-python-simplifying-environment-management-securing-your-secrets-14c51c411400> [Accessed 09 Oct. 2025].
4. Akamai. (2025). *What Is API Authorization? Akamai*. [online] Available at: <https://www.akamai.com/glossary/what-is-api-authorization> Akamai. (2025). *What Is API Authorization? Akamai*. [online] Available at: <https://www.akamai.com/glossary/what-is-api-authorization> [Accessed 09 Oct. 2025].