Cyber Security Internship – Task 4
Report & Findings

Name: Abdullah Akram

Roll No: DHC-3606

Week: 4

Title: Advanced Threat Detection & Web Security Enhancements

July 13, 2025

1 Overview

In Week 4 of the Cyber Security Internship, the primary objective was to enhance the security of a PHP-based login system. The focus was on implementing advanced security measures to protect API endpoints, detect threats in real-time, and enforce strict security headers to mitigate common web vulnerabilities. The following sections detail the implemented enhancements, their functionality, and their impact on the system's resilience.

2 API Security Hardening

To secure the API endpoints, several measures were implemented to ensure only authorized access and to prevent abuse:

- **API Key Authentication**: An API key-based authentication mechanism was integrated. The key, securely stored in the `config.php` file as `API_SECRET_KEY`, is validated via the `X-API-KEY` header in `login.php`. Unauthorized requests without a valid key receive a 401 Unauthorized response, ensuring only trusted clients can access the API.
- **Rate Limiting**: To prevent brute-force attacks, rate limiting was implemented, restricting each IP address to 5 login attempts per minute. This is achieved using per-IP JSON files stored in a dedicated `rates/` folder. Exceeding the limit triggers an HTTP 429 (Too Many Requests) response, effectively blocking abusive behavior.
- **CORS Configuration**: Cross-Origin Resource Sharing (CORS) headers were configured to restrict access to trusted origins only. For development, the `Access-Control-Allow-Origin` is set to `*`, but it will be updated to the production domain to prevent unauthorized cross-origin requests.

3 Security Headers Implementation

To further strengthen the application against common web threats, the following security headers were added to `login.php`:

- **Content Security Policy (CSP)**: A strict CSP was implemented to prevent script injections. The policy restricts resources to `self`, allowing only scripts and styles from the same origin, with `unsafe-inline` permitted for styles to maintain compatibility with existing CSS.
- **Strict-Transport-Security (HSTS)**: HSTS headers were added to enforce HTTPS connections, ensuring all communications are encrypted. The header includes a `max-age` of 2 years and applies to all subdomains, enhancing transport layer security.
- **Additional Headers**: Headers such as `X-Content-Type-Options: nosniff`, `X-Frame-Options: DENY`, and `X-XSS-Protection: 1; mode=block` were included to prevent MIME-type sniffing, clickjacking, and enable browser-level XSS protection, respectively.

4 Testing and Validation

The implemented security measures were thoroughly tested to ensure functionality:

- Invalid API key requests correctly return a 401 Unauthorized response.
- Excessive login attempts (more than 5 within a minute) result in a 429 Too Many Requests error.
- Valid credentials successfully authenticate users, set session variables, and redirect to `dashboard.php`, confirming a secure login flow.

5 Conclusion

The enhancements implemented in Week 4 have significantly hardened the PHP-based login system against common web threats. The API is now protected with key-based authentication, rate limiting, and CORS restrictions, while security headers like CSP and HSTS provide robust protection against injections and insecure connections. The system is production-ready and resilient, with potential for further improvements such as token-based authentication (JWT) and Linux-level intrusion detection tools in future iterations. All internship guidelines for Task 4 have been successfully met.