Web Application Vulnerabilities

• **Unrestricted File Upload**: The file upload feature allows uploading and executing PHP files, enabling an attacker to run arbitrary server-side commands within the /uploads/ directory.

• **Improper Access Control – Unauthorized File Download**: The download_file.php endpoint does not verify ownership or permissions, allowing users to download files from vaults they do not have access to.

• **Improper Access Control – Direct Access to Vault Details**: Vault details pages can be accessed directly via URL without session validation, exposing sensitive vault content to unauthenticated users.

• **Hard-Coded Database Credentials**: Multiple PHP source files contain plaintext MySQL database credentials, allowing anyone with source access to log in directly to the database.

• **Cross-Site Request Forgery (CSRF) on Privileged Operations**: Administrative actions such as user approval and permission assignments lack CSRF protection, enabling attackers to escalate privileges through crafted requests.

• **Missing Brute Force Protection**: The login page allows unlimited credential attempts, making it vulnerable to dictionary or automated brute-force login attacks.

• **Session Does Not Expire / Persistent Session**: User sessions remain valid indefinitely even after browser closure or inactivity, allowing unauthorized reuse of authenticated sessions.

• **Stored Cross-Site Scripting (Stored XSS)**: User-controlled input in the vault creation form is not sanitized, allowing malicious script execution within other users' browsers via stored payloads.

• **Insecure Direct Object Reference (IDOR)**: Changing the vault_id parameter in URLs exposes vaults belonging to other users, bypassing authorization checks.

• **Reflected Cross-Site Scripting**: Certain pages reflect unvalidated user input back to the browser, allowing attackers to inject and execute scripts dynamically.

• **Path Traversal**: File path parameters can be manipulated to access unintended files by including directory traversal sequences like ../.

• **SQL Injection**: Dynamic SQL queries built with unsanitized PHP user input allow attackers to modify or extract data from the MySQL database.

• **Missing Anti-CSRF Tokens**: Sensitive pages contain forms lacking CSRF tokens, making them susceptible to unauthorized cross-origin form submissions.

• **Content Security Policy (CSP) Not Implemented**: No CSP header is set, allowing unrestricted inline script execution and increasing XSS attack impact.

• **Missing Anti-Clickjacking Header**: The application can be embedded within an iframe, enabling clickjacking attacks.

• **Cookies Missing Security Flags**: Session cookies are missing HttpOnly, Secure, and SameSite flags, making them vulnerable to interception and misuse.

• **Information Disclosure via Response Headers**: Response headers reveal server software and framework details, aiding reconnaissance for targeted exploitation.

• **Strict-Transport-Security (HSTS) Not Set**: The application does not enforce HTTPS, allowing downgrade attacks and potential session hijacking.

• **Missing X-Content-Type-Options Header**: Browsers can MIME-sniff responses, increasing risk for drive-by script execution.

• **Authentication Request Exposed**: Authentication endpoints may leak identifiable patterns useful for attacker reconnaissance.

• **Loosely Scoped Cookie**: Cookies are accessible beyond necessary path or scope, increasing attack surface.

• **Potentially Misconfigured Cache-Control**: Sensitive authenticated pages may be cached, enabling local recovery of private data.

• **Insecure Session Management Behaviors Observed**: Session handling lacks modern security controls such as regeneration, invalidation on logout, and binding to device context.