**Insecure Blog Application**

1. **Application Structure**

**Core Components**

|  |  |  |
| --- | --- | --- |
| **File** | **Purpose** | **Security Status** |
| config.php | Database connection & security headers | Secure |
| login.php | User authentication | Vulnerable (SQLi) |
| profile.php | User profile display | Vulnerable (XSS) |
| search.php | Blog post search | Secure |
| upload.php | File upload & URL fetching | Vulnerable (SSRF) |
| admin.php | Admin dashboard | Secure |
| post.php | Blog post display | Secure |
| home.html | Homepage (DOM XSS removed) | Secure |

**Data Flow**

1. User Authentication (login.php) → SQL query executed.
2. Profile Display (profile.php) → Renders user data with potential XSS.
3. File Handling (upload.php) → Allows arbitrary URL fetching (SSRF).
4. **Vulnerability Report (Other Team’s Website)**

**Found Vulnerabilities**

|  |  |  |
| --- | --- | --- |
| **Vulnerability** | **Exploitation Method** | **Impact** |
| **SQL Injection** | Inject ' OR 1=1 -- in login form | Bypass authentication, dump database |
| **XSS (Stored/DOM)** | Submit <script>alert(1)</script> in comments | Session hijacking, defacement |

**Exploitation Demo**

1. SQLi Attack
   * Input: admin' -- in username field → Bypasses password check.
   * Exploit: ahmed@gmail.com' OR '1'='1--
   * admin@gmail.com'--
2. XSS Attack
   * <h1>hello</h1>
3. Csrf

* <http://localhost:8080/cyberproj/api.php?endpoint=register>

**3. Security Improvements (Patching)**

**Fixed Vulnerabilities**

|  |  |  |
| --- | --- | --- |
| Vulnerability | Patch Method | Code Example |
| SQL Injection | Use prepared statements | $stmt = $pdo->prepare("SELECT \* FROM users WHERE username = ?"); |
| XSS | Apply htmlspecialchars() | <?= htmlspecialchars($user\_input) ?> |
| SSRF | Restrict URL fetching | if (!in\_array(parse\_url($url, PHP\_URL\_HOST), ['trusted.com'])) die(); |

**Example Fixes**

1. **SQLi Patch (login.php)**

$stmt = $pdo->prepare("SELECT \* FROM users WHERE username = ? AND password = ?");

$stmt->execute([$username, $password]);

Never concatenate inputs into the query

Always pass parameters via execute

1. **XSS Patch (profile.php)**

<div><?= htmlspecialchars($\_GET['greeting']) ?></div>

The browser displays the code as plain text instead of executing it.

1. **SSRF Patch (upload.php)**

**Use allowlists** (only permit trusted domains):

Php:

$allowed = ['trusted.com', 'api.safe.org'];

if (!in\_array($parsed['host'], $allowed)) {

die("URL not allowed!");

}