Lawyer Portal System

Threat Modeling

1. **Critical Assets Identification**

Core Components:

- MySQL Database (Cases & Users Storage)
- PHP User Interface
- Session Management System
- Role-Based Access Control (RBAC)
- Internal Case Management APIs

Sensitive Data:

- User Credentials (Names, Roles, Permissions)
- Case Records (Title, Status, Timestamps)
- Session IDs

2. **Threat Identification Using STRIDE Model**

Category	Potential Threats	Project Example
Spoofing	User identity theft via session hijacking	Stolen session IDs from insecure cookies
Tampering	Unauthorized data modification	SQL Injection altering case status
Repudiation	Denial of executed actions	Missing audit logs for delete operations
Information Disclosure	Sensitive data exposure	Unauthorized case data exposure
DoS	Service disruption via resource exhaustion	Login page brute-force attacks
Elevation	Privilege escalation vulnerabilities	URL manipulation to change user roles

3. **Vulnerability & Attack Surface Analysis**

Component	Vulnerability	Attack Vector	Severity
Authentication	Weak passwords	login.php	High
Session Management	Session fixation	URL-exposed session IDs	Medium
SQL Queries	SQL Injection	Case add/edit endpoints	Critical
RBAC System	IDOR Vulnerability	URL parameter manipulation	Medium
Configuration Files	Directory traversal	Access to config.php	High

4. Risk Assessment Using DREAD Model

Threat	Damage	Reproducibility	Exploitability	Affected Users	Discoverability	Total
SQL Injection	5	4	3	5	4	21
Session Hijacking	4	3	2	4	3	16
Privilege Escalation	5	2	2	3	2	14
Data Exposure	3	4	3	4	3	17

5. Mitigation Strategies

A. SQL Injection Prevention

```
$stmt = $conn->prepare("SELECT * FROM cases WHERE id = :id");
$stmt->bindParam(':id', $id, PDO::PARAM_INT);
$stmt->execute();
```

B. Secure Session Management

```
session.cookie_httponly = 1
session.cookie_secure = 1
session.use_strict_mode = 1
session_regenerate_id(true);
```

C. Enhanced RBAC Implementation

```
function checkPermission($required) {
    if (!isset($_SESSION['user_role'])) {
        error_log("Unauthorized access attempt from IP:
".$_SERVER['REMOTE_ADDR']);
        header("Location: /unauthorized.php");
        exit;
    }
    if (!has_permission($_SESSION['user_role'], $required)) {
        error_log("Permission denied for user:
".$_SESSION['user_id']);
        throw new AccessDeniedException();
    }
}
```

D. Common Attack Protections

```
<IfModule mod_headers.c>
    Header set X-XSS-Protection "1; mode=block"
    Header set Strict-Transport-Security "max-age=63072000"
    Header set Content-Security-Policy "default-src 'self'"
</IfModule>
```

E. Additional Security Measures

- Encryption: AES-256 for sensitive data at rest
- Backups: Daily encrypted database backups
- Patching: Immediate security updates implementation

6. Data Flow Diagram

[User] --> [Login] --> (Session ID) --> [RBAC] --> [Case Mgmt] --> [DB] [DB] <--> [Encrypted Backup]

7. **Incident Response Plan**

Phase	Actions
Detection	Monitor failed login attempts & unauthorized access patterns
Assessment	Analyze logs for IOC patterns
Containment	Isolate affected systems/disable compromised accounts
Eradication	Remove malicious code/apply security patches
Recovery	Restore from clean backups
Post-Mortem	Update security policies/conduct team training

8. **Recommended Tools**

- Code Analysis: SonarQube, PHPStan
- Pen Testing: OWASP ZAP
- Server Hardening: Lynis
- Log Analysis: Graylog (Basic)

9. **Final Recommendations**
10. **Implement Two-Factor Authentication for admin accounts**
11. **Conduct quarterly security audits**
12. **Enforce least privilege principle for all users**
13. **Add CSP headers to prevent XSS attacks**
14. **Maintain access logs for 90+ days.**