Задание 7

Уязвимости XSS

- **Описание**: Уязвимость из-за вывода данных без экранирования.
- **Проблемы**: Данные (fio, biography) в index.php и admin.php выводились как `echo $app['fio'];`.
- **Защита**: Использование `htmlspecialchars()` и Content Security Policy (CSP).
- **Пример**:

```
header("Content-Security-Policy: default-src 'self';");
```

Information Disclosure

- **Описание**: Раскрытие ошибок или данных.
- **Проблемы**: Ошибки PDO в config.php и отладочные echo в form.php.
- **Защита**: Логирование ошибок, удаление echo.
- **Пример**:

```
error_log($e->getMessage());
exit("Ошибка сервера.");
```

SQL Injection

- **Описание**: Манипуляция запросами. Возможность выполнения произвольных SQL-запросов через ввод пользователя.
- **Проблемы**: Подготовленные запросы без отключения эмуляции. Пример уязвимого запроса: `SELECT * FROM users WHERE email = '".$_POST['email']."'`.
- **Защита**: Отключение эмуляции, использование prepare. Использование подготовленных выражений (PDO).
- **Пример**:

```
$pdo->setAttribute ATTR_EMULATE_PREPARES, false ; // Отключение эмуляции
$stmt = $pdo->prepare("SELECT * FROM users WHERE email = ?"); //
Подготовленное выражение
$stmt->execute([$value]); // или [$_POST['email']]
```

CSRF

- **Описание**: Запросы от имени пользователя. Подделка запроса от имени пользователя.
- **Проблемы**: Отсутствие токена, удаление через GET. Отсутствие проверки CSRF-токенов в form.php.
- **Защита**: Добавление CSRF-токена, использование POST. Генерация и проверка токена в сессии.
- **Пример**:
  - Генерация: `$_SESSION['csrf_token'] = bin2hex(random_bytes(32));`
  - Форма:

    ```
    <input type="hidden" name="csrf_token" value="<?php echo
    $_SESSION['csrf_token']; ?>">
    ```

    или

    ```
    <input type="hidden" name="csrf" value="<?php echo
    $_SESSION['token']; ?>">
    ```

  - Проверка:

    ```
    if (!hash_equals($_SESSION['csrf_token'], $_POST['csrf_token']))
        die("Недействительный токен.");
    ```

Include

- **Описание**: Включение вредоносных файлов. Возможность подключения произвольных файлов.
- **Проблемы**: Отсутствие проверки пути в require_once. `include $_GET['page'].".php";`.
- **Защита**: Использование абсолютных путей и проверки. Белый список файлов или жёсткое задание.
- **Пример**:
  - Использование абсолютных путей и проверки существования файла:

    ```
    $configFile = __DIR__ . '/config.php';
    if (file_exists($configFile))
        require_once $configFile;
    ```

```
else
    die("Файл не найден.");
```

- Использование белого списка:

```
$allowed = ['main', 'about'];
if (in_array($_GET['page'], $allowed)) {
    include $_GET['page'].".php";
}
```

Upload

- **Описание**: Загрузка вредоносных файлов.
- **Проблемы**: Отсутствует механизм загрузки. Отсутствие валидации MIME-типа и расширения.
- **Защита**: Проверка типов и размера файлов. Проверка расширения, MIME-типа, генерация случайного имени.
- **Пример**:
  - Проверка типа (MIME) и размера, генерация случайного имени с фиксированным расширением:

```
if ($_FILES['avatar']['error'] == UPLOAD_ERR_OK) {
    $allowed = ['image/jpeg', 'image/png'];
    if (!in_array(mime_content_type($_FILES['avatar']['tmp_name']),
$allowed) || $_FILES['avatar']['size'] > 2000000) {
        $errors[] = 'Недопустимый файл.';
    } else {
        $filename = uniqid() . '.jpg'; // Обратите внимание:
фиксированное расширение '.jpg' независимо от MIME типа
        move_uploaded_file($_FILES['avatar']['tmp_name'], 'uploads/'
. $filename);
    }
}
```

  - Проверка расширения и генерация случайного имени с сохранением исходного расширения:

```
$ext = pathinfo($_FILES['file']['name'], PATHINFO_EXTENSION);
if (in_array($ext, ['jpg','png'])) {
    move_uploaded_file($_FILES['file']['tmp_name'],
```

```
"uploads/".uniqid().".$ext");
}
```