Защита от Information Disclosure

Каждому пользователю назначена своя специальная роль. Для администратора - это статус "admin", для обычного пользователя - статус "user". При входе в панель выполняется проверка прав доступа пользователя.

```
function checkAdmin($login, $password){
      global $db;
      if(isset($login) && isset($password)){
          $qu = $db->prepare("SELECT id FROM users WHERE role = 'admin' and
login = ? and password = ?");
          $qu->execute([$login, md5($password)]);
          return $qu->rowCount();
      }
}
```

Так же при открытии страницы формы, проверяется на то, что пользователь авторизован или нет. Если пользователь не авторизован, выводится пустая форма. Если пользователь авторизован, тогда из базы данных ищутся данные формы, связанные с этим пользователем. Если такие данные не найдены, выполняется следующая функция:

```
  function user_exit(){
    del_cook_all(1);
    session_destroy();
    header('Location: index.php');
    exit();
  }
```

Эта функция выполняет logout пользователя, удаляет все cookie и выводит пустую форму. Также использовано ini_set(), чтобы не выводить на экран ошибки:

```
ini_set('display_errors', 'off');
```

Защита от CSRF

Проверяется совпадение подлинности запроса с данными, которые хранятся в сессии:

```
Генерируем csrf_token, сохраняем в сессию и выводим в форму
<?php
      $csrf_token = bin2hex(random_bytes(32));
      $_SESSION['csrf_token'] = $csrf_token;
?>
<input type="hidden" name='csrf_token' value='<?php echo $csrf_token; ?>'>
```

```
При поступлении запроса на сервер, получаем из сессии и из POST csrf_token и
сравниваем
$csrf_error = (isset($_COOKIE['csrf_error']) ? checkInput($_COOKIE['csrf_error']) :
'');

if (isset($_COOKIE['csrf_error'])) {
      $messages['error'] = 'Не соответствие CSRF токена';
      del_cook('csrf');
```

Данная проверка применяется при отправке форм, процедуре авторизации и запросах из панели администратора. Помимо этого, для защиты от атак CSRF используются дополнительные меры:

```
    ini_set('session.cookie_samesite', 'Lax');
```

Защита от Include уязвимости

В php не подключаются сторонние файлы, поэтому данном коде эта проверка не реализована.

В случае, если было бы подключение, нужно было бы использовать белый список разрешенных файлов, который в процессе выполнения кода проверялся.

Защита от XSS

Данная функция используется при получении данных из базы данных, чтобы предотвратить внедрение js скриптов. Она убирает лишние пробелы в начале и в конце, удаляет html тэги и преобразует в html объекты и символы.

```
    function checkInput($param){
        return htmlspecialchars(strip_tags(trim($param)), ENT_QUOTES);
    }

Пример использования:
    $fio = (isset($_POST['fio']) ? checkInput($_POST['fio']) : '');
```

Защита от SQL Injection

Используется PDO (подготовленные запросы) и дополнительно вызывается функция checkInput (выше) для дополнительной защиты.

```
        $stmt = $db->prepare("INSERT INTO users (login, password) VALUES (?,?)");
        $stmt->execute([$login, $mpassword]);
```

Защита от Upload уязвимости

Т.к. в форме обратной связи загрузка файлов не предусмотрена, то данная проверка не была реализована.

В случае, если бы файлы загружались, нужно было бы проверить тип файла, его размер (ограничить макс размер на стороне сервера) и запретить выполнение скриптов в каталоге, где находились бы загруженные файлы.