Кощавцев Даниил.21/1

Аудит безопасности

1.1. Уязвимости XSS:

– Уязвимость передачи JS скрипта

Чтобы предотвратить передачу данных, которые не удовлетворяют условиям, я ввёл проверку на соответствие передаваемых данных запрошенным и завалидовал переменные, использовал strip_tags(), htmlspecialchars(), preg_match(), а также доп. Проверки данных:

```
$errors = FALSE;
if (empty($_POST['fio']) || (!preg_match($nameregex,$_POST['fio'])) ) {
  // Выдаем куку на день с флажком об ошибке в поле fio.
  setcookie('fio_error', '1', time() + 24 * 60 * 60);
  setcookie('fio_value', '', 100000);
  $errors = TRUE;
}

if (empty($_POST['email']) || !filter_var($_POST['email'], FILTER_VALIDATE_EMAIL) || !preg_match($mailregex,$_POST['email'])) {
  // Выдаем куку на день с флажком об ошибке в поле fio.
  setcookie('email_error', '1', time() + 24 * 60 * 60);
  setcookie('email_value', '', 100000);
  $errors = TRUE;
}

if ($_POST['year']=='Не выбран') {
  // Выдаем куку на день с флажком об ошибке в поле fio.
  setcookie('year_error', '1', time() + 24 * 60 * 60);
  setcookie('year_value', '', 100000);
  $errors = TRUE;
}

if (!isset($_POST['sex'])) {
  // Выдаем куку на день с флажком об ошибке в поле fio.
  setcookie('sex_error', '1', time() + 24 * 60 * 60);
  setcookie('sex_value', '', 100000);
  $errors = TRUE;
}
```

```
try {
  $id=$_GET['edit_id'];
  $get=$db->prepare("SELECT * FROM app WHERE id=?");
  $get->bindParam(1, $id);
  $get->execute();
  $info=$get->fetchALL();
  $values['fio']=htmlspecialchars(strip_tags($info[0]['name']));
  $values['email']=htmlspecialchars(strip_tags($info[0]['email']));
  $values['year']=htmlspecialchars(strip_tags($info[0]['year']));
  $values['sex']=htmlspecialchars(strip_tags($info[0]['sex']));
  $values['limbs']=htmlspecialchars(strip_tags($info[0]['limbs']));
  $values['biography']=htmlspecialchars(strip_tags($info[0]['biography']));
```

```
<div class = "name"><label>FIO</label>
<input name= "fio" <?php if ($errors['fio']) {print 'class="error"';} ?> value="<?php print strip_tags($values['fio']); ?>" /></div>
 <div class = "email"><label>E-MAIL</label>
 <input name="email" <?php if ($errors['email']) {print 'class="error"';} ?> value="<?php print strip_tags($values['email']); ?>" /></div>
 <div class = "year"><label>YEAR</label>
 <select name="year" <?php if ($errors['year']) {print 'class="error"';} ?> value="<?php print strip_tags($values['year']); ?>">
 <?php
```

1.2. Уязвимости SQL Injection:

Для предотвращения уязвимостей SQL Injection я использовал подготовленные запросы (PDO) и проверял данные на соответствие с помощью функции filter_var() и is_numeric():

```
<div class = "name"><label>FIO</label>
<input name= "fio" <?php if ($errors['fio']) {print 'class="error"';} ?> value="<?php print strip_tags($values['fio']); ?>" /></div>
 <div class = "email"><label>E-MAIL</label>
 <input name="email" <?php if ($errors['email']) {print 'class="error"';} ?> value="<?php print strip_tags($values['email']); ?>" /></div>
 <div class = "year"><label>YEAR</label>
 <select name="year" <?php if ($errors['year']) {print 'class="error"';} ?> value="<?php print strip_tags($values['year']); ?>">
 <?php
$errors = FALSE;
if (empty($_POST['fio']) || (!preg_match($nameregex,$_POST['fio'])) ) {
  // Выдаем куку на день с флажком об ошибке в поле fio.
  setcookie('fio_error', '1', time() + 24 * 60 * 60);
  setcookie('fio_value', '', 100000);
  $errors = TRUE;
}

if (empty($_POST['email']) || !filter_var($_POST['email'], FILTER_VALIDATE_EMAIL) || !preg_match($mailregex,$_POST['email'])) {
  // Выдаем куку на день с флажком об ошибке в поле fio.
  setcookie('email_error', '1', time() + 24 * 60 * 60);
  setcookie('email_value', '', 100000);
  $errors = TRUE;
}

if ($_POST['year']=='Не выбран' || !(is_numeric($_POST['year']))) {
  // Выдаем куку на день с флажком об ошибке в поле fio.
  setcookie('year_error', '1', time() + 24 * 60 * 60);
  setcookie('year_value', '', 100000);
  $errors = TRUE;
}

if (!isset($_POST['sex']) || !(is_numeric($_POST['sex']))) {
  // Выдаем куку на день с флажком об ошибке в поле fio.
  setcookie('sex_error', '1', time() + 24 * 60 * 60);
  setcookie('sex_value', '', 100000);
  $errors = TRUE;
}

if (!isset($_POST['limbs']) || !(is_numeric($_POST['limbs']))) {
  // Выдаем куку на день с флажком об ошибке в поле fio.
  setcookie('limbs_error', '1', time() + 24 * 60 * 60);
  setcookie('limbs_value', '', 100000);
  $errors = TRUE;
}
```

```
    $id=$_POST['nform'];
    $user = 'u52997';
    $pass = '4390881';
    $db = new PDO('mysql:host=localhost;dbname=u52997', $user, $pass, array(PDO::ATTR_PERSISTENT => true));
    try {
      $del=$db->prepare("DELETE FROM app_abil WHERE aplication_id=?");
      $del->execute(array($id));
      $stmt = $db->prepare("DELETE FROM app WHERE id=?");
      $stmt -> execute(array($id));
    }
   catch(PDOException $e){
    print('Error : ' . $e->getMessage());
    exit();
   }
```

```
$user = 'u52997';
$pass = '4390881';
$db = new PDO('mysql:host=localhost;dbname=u52997', $user, $pass, array(PDO::ATTR_PERSISTENT => true));

if (!$errors) {
  $upd=$db->prepare("UPDATE app SET name=:name, email=:email, year=:year, sex=:sex, limbs=:limbs, biography=:biography WHERE id=:id");
  $cols=array(
    ':name'=>$name,
    ':email'=>$email,
    ':year'=>$year,
    ':sex'=>$sex,
    ':limbs'=>$limbs,
    ':biography'=>$biography
  );
  foreach($cols as $ind=>&$val){
    $upd->bindParam($ind,$val);
  }
    $upd->bindParam(':id',$id);
    $upd->execute();
    $del=$db->prepare("DELETE FROM app_abil WHERE aplication_id=?");
    $del->execute(array($id));
    $upd1=$db->prepare("INSERT INTO app_abil SET ability=:ability, aplication_id=:id");
    $upd1->bindParam(':id',$id);
    foreach($ability as $abl){
      $upd1->bindParam(':ability',$abl);
      $upd1->execute();
    }
  }
```

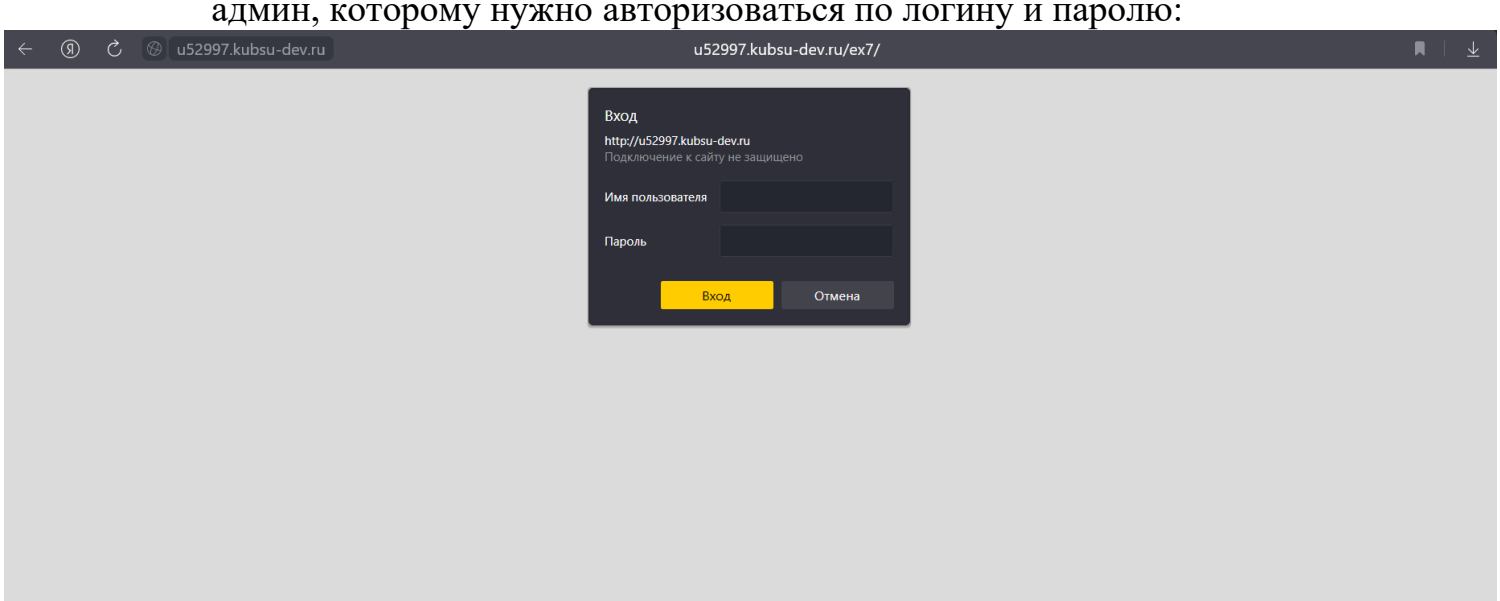
1.3. Уязвимости CSRF

Для защиты от уязвимости CSRF я по назначению использовал методы POST и GET, а также реализовал устаревание куки-файлов:

```php
header('Content-Type: text/html; charset=UTF-8');
// В суперглобальном массиве $_SERVER PHP сохраняет некторые заголовки запроса HTTP
// и другие сведения о клиненте и сервере, например метод текущего запроса $_SERVER['REQUEST_METHOD'].
if ($_SERVER['REQUEST_METHOD'] == 'GET') {
  // Массив для временного хранения сообщений пользователю.
  $messages = array();

  // В суперглобальном массиве $_COOKIE PHP хранит все имена и значения куки текущего запроса.
  // Выдаем сообщение об успешном сохранении.
  if (!empty($_COOKIE['save'])) {
    setcookie('save', '', 100000);

    // Выводим сообщение пользователю.
    $messages[] = 'Спасибо, результаты сохранены.';

    setcookie('name_value', '', 100000);
    setcookie('email_value', '', 100000);
    setcookie('year_value', '', 100000);
    setcookie('sex_value', '', 100000);
    setcookie('limbs_value', '', 100000);
    setcookie('biography_value', '', 100000);
    setcookie('ab_in_value', '', 100000);
    setcookie('ab_t_value', '', 100000);
    setcookie('ab_l_value', '', 100000);
    setcookie('ab_v_value', '', 100000);
    setcookie('check_value', '', 100000);
  }
```

1.4. Include и Upload уязвимости

В моём коде не использовались фреймворки, общий доступ имеет только админ, которому нужно авторизоваться по логину и паролю:



А также я запретил реализацию скриптов на сайте, использовав настройку файлом .htaccess, который я поместил в папку с файлами table и form:

.htaccess

```
RemoveHandler .php
RemoveType .php
AddType application/x-httpd-php-source .php
Options -ExecCGI -Indexes
php_flag engine off
```

Имя файла	Размер	Тип фай...	Последнее...	Права	Владеле...
..					
.htaccess	128	Файл "H...	16.05.2023 ...	-rw-r--r--	u52997 w...
form.php	3 701	Файл "P...	19.05.2023 ...	-rw-r--r--	u52997 w...
table.php	2 816	Файл "P...	19.05.2023 ...	-rw-r--r--	u52997 w...