Отчёт по безопасности. Проверка и исправление ошибок, связанных с уязвимостями.

XSS

В коде не было никакой фильтрации или экранирования выходных данных. Также в коде не было проверки на наличие недопустимых символов в входных данных.

Код до:

```
if (empty($_POST['fio']) || !preg_match('/^[а-яА-ЯёЁa-zA-Z\s-]{1,150}$/u',
$_POST['fio'])) {
  print ('Заполните имя.<br/>');
  $errors = TRUE;
}
if (empty($_POST['tel']) || !preg_match('/^\+[0-9]{11}$/', $_POST['tel'])) {
  print ('Заполните телефон.<br/>');
  $errors = TRUE;
}
```

Код после:

```
if (empty($_POST['fio']) || !preg_match('/^[а-яА-ЯёЁa-zA-Z\s-]{1,150}$/u',
$_POST['fio'])) {
  print (htmlspecialchars('Заполните имя.<br/>'));
  $errors = TRUE;
}
if (empty($_POST['tel']) || !preg_match('/^\+[0-9]{11}$/', $_POST['tel'])) {
  print (htmlspecialchars('Заполните телефон.<br/>'));
  $errors = TRUE;
}
```

И проверки добавленные:

```
if (empty($_POST['email']) || !filter_var($_POST['email'],
FILTER_VALIDATE_EMAIL)) {
  $errors = TRUE;
}

// Проверяем, что дата рождения содержит только цифры и разделена точками
if (empty($_POST['day']) || !preg_match('/^\d{1,2}$/', $_POST['day']) ||
    empty($_POST['month']) || !preg_match('/^\d{1,2}$/', $_POST['month']) ||
    empty($_POST['year']) || !preg_match('/^\d{4}$/', $_POST['year'])) {
  $errors = TRUE;
```

```
}
```

Для устранения уязвимости добавил фильтрацию и экранирование выходных данных. Для этого использовал функцию htmlspecialchars(). Также добавил проверку на наличие недопустимых символов в входных данных, использовав для это регулярные выражения.

Information Disclosure

В коде до была ошибка, связанная с тем что, что данные для подключения к базе данных хранятся в виде обычного текста в файле index.php.

Код до:

```
$user = 'u67345';
$pass = '2030923';
$db = new PDO(
  'mysql:host=localhost;dbname=u67345',
  $user,
  $pass,
  [PDO::ATTR_PERSISTENT => true, PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]
);
```

Код после:

```
 include('../password.php');
```

Создал отдельный файл, вынес в него пароль, логин и подключение к базе данных, внёс его в gitignore и добавил его напрямую на сервер.

SQL Injection

В коде до была уязвимость связанная с тем что, что значение **$_POST['lang']** используется в SQL-запросе без какого-либо экранирования или фильтрации.

Код до:

```
  $stmt = $db->prepare("INSERT INTO person_lang (id_u, id_l) VALUES
(:id_u,:id_l)");
  foreach ($_POST['lang'] as $lang) {
    $stmt->bindParam(':id_u', $id_u);
    $stmt->bindParam(':id_l', $lang);
```

```
    $id_u=$id;
    $stmt->execute();
  }
```

Код после:

```
  foreach ($_POST['lang'] as $lang) {
    $lang = $db->quote($lang);
    $stmt->bindParam(':id_u', $id_u);
    $stmt->bindParam(':id_l', $lang);
    $id_u=$id;
    $stmt->execute();
  }
```

Для исправления я использовал строчку `$lang = $db->quote($lang);`, То есть экранировал значение $_POST['lang'] перед использованием его в SQL-запросе.

CSRF

В этом коде отсутствовала защита от CSRF-атак. Это означает, что злоумышленник может подделать запрос на изменение пароля или другое действие, которое требует авторизации, и выполнить его от имени пользователя, который в настоящее время авторизован в приложении.

Код до:

```
if ($_SERVER['REQUEST_METHOD'] == 'GET') {
?>

<form action="" method="post">
  <input name="login" />
  <input name="pass" />
  <input type="submit" value="Войти" />
</form>

<?php
}
```

```
<form action="" method="post">
  <input name="login" />
  <input name="pass" />
  <input type="submit" value="Войти" />
</form>

<?php
}
else {
  include('../password.php');
  $login = $_POST['login'];
```

```
  $pass = md5($_POST['pass']);
  $sth = $db->prepare("SELECT*FROM person_login");
  $sth->execute();
  $log_pass = $sth->fetchAll();
  $error_logpas = true;
  foreach($log_pass as $lp){
    if($login == $lp['login_u'] && $pass == $lp['pass_u']){
      $error_logpas = false;
      break;
    }
  }
  if($error_logpas == true){
    print('<div> Ошибка пользователя с таким логином или паролем нет </div>');
  }else{
  if (!$session_started) {
    print('<div> Всё в порядке </div>');
    session_start();
  }
}
  $_SESSION['login'] = $_POST['login'];
  $_SESSION['uid'] = count($log_pass);
  header('Location: ./');
}
```

Код после:

```
if ($_SERVER['REQUEST_METHOD'] == 'GET') {
    // Генерируем токен безопасности для защиты от CSRF-атак
    $csrf_token = bin2hex(random_bytes(32));
    // Сохраняем токен в сессию
    $_SESSION['csrf_token'] = $csrf_token;
    ?>

    <form action="" method="post">
      <input name="login" />
      <input name="pass" />
      <!-- Добавляем скрытое поле с токеном безопасности -->
      <input type="hidden" name="csrf_token" value="<?php echo $csrf_token; ?>"
/>
      <input type="submit" value="Войти" />
    </form>

    <?php
}
else {
    include('../password.php');
    $login = $_POST['login'];
    $pass = md5($_POST['pass']);
    $sth = $db->prepare("SELECT*FROM person_login");
```

```
    $sth->execute();
    $log_pass = $sth->fetchAll();
    $error_logpas = true;
    foreach($log_pass as $lp){
        if($login == $lp['login_u'] && $pass == $lp['pass_u']){
            $error_logpas = false;
            break;
        }
    }
    if($error_logpas == true){
        print('<div> Ошибка пользователя с таким логином или паролем нет
</div>');
    }else{
        if (!$session_started) {
            print('<div> Всё в порядке </div>');
            session_start();
        }
        // Проверяем токен безопасности
        if (!isset($_POST['csrf_token']) || $_POST['csrf_token'] !=
$_SESSION['csrf_token']) {
            print('<div> Ошибка токена безопасности </div>');
        } else {
            // Если все ок, то авторизуем пользователя.
            $_SESSION['login'] = $_POST['login'];
            // Записываем ID пользователя.
            $_SESSION['uid'] = count($log_pass);

            // Делаем перенаправление.
            header('Location: ./');
        }
    }
}
?>
```

Чтобы исправить эту уязвимость, я добавил в форму скрытое поле с токеном безопасности, который будет генерироваться для каждого пользователя и каждого сеанса, и проверять этот токен на сервере при обработке запроса.

Include

Так как в файлах index.php я не добавляю сторонние файлы и не предоставляю такой возможности пользователям, имею белый список файлов, то уязвимости связанной с Include у меня нет.

```
<?php
header('Content-Type: text/html; charset=UTF-8');
```

```php
if ($_SERVER['REQUEST_METHOD'] == 'GET') {
  if (!empty ($_GET['save'])) {
    print (htmlspecialchars('Спасибо, результаты сохранены.'));
  }
  include ('form.php');
  exit();
}
$errors = FALSE;
if (empty($_POST['fio']) || !preg_match('/^[a-яА-ЯёЁa-zA-Z\s-]{1,150}$/u',
$_POST['fio'])) {
  print (htmlspecialchars('Заполните имя.<br/>'));
  $errors = TRUE;
}
if (empty($_POST['tel']) || !preg_match('/^\+[0-9]{11}$/', $_POST['tel'])) {
  print (htmlspecialchars('Заполните телефон.<br/>'));
  $errors = TRUE;
}

if (empty($_POST['fio']) || !preg_match('/^[a-zA-Za-яА-ЯёЁ\s]+$/u',
$_POST['fio'])) {
  $errors = TRUE;
}

// Проверяем, что телефон содержит только цифры и начинается с плюса
if (empty($_POST['tel']) || !preg_match('/^\+\d{11}$/', $_POST['tel'])) {
  $errors = TRUE;
}

// Проверяем, что почта соответствует формату email
if (empty($_POST['email']) || !filter_var($_POST['email'],
FILTER_VALIDATE_EMAIL)) {
  $errors = TRUE;
}

// Проверяем, что дата рождения содержит только цифры и разделена точками
if (empty($_POST['day']) || !preg_match('/^\d{1,2}$/', $_POST['day']) ||
    empty($_POST['month']) || !preg_match('/^\d{1,2}$/', $_POST['month']) ||
    empty($_POST['year']) || !preg_match('/^\d{4}$/', $_POST['year'])) {
  $errors = TRUE;
}

// Проверяем, что пол соответствует одному из двух вариантов
if (empty($_POST['gender']) || ($_POST['gender'] != 'man' && $_POST['gender'] !=
'woman')) {
  $errors = TRUE;
}
if (empty($_POST['bio']) || !preg_match('/^[a-zA-Za-яА-ЯёЁ0-9\s.,!?:;-]+$/u',
$_POST['bio'])) {
  $errors = TRUE;
}
```

```php
include('../password.php');

if (empty($_POST['like-4'])) {
  $errors = TRUE;
} else {
  $sth = $db->prepare("SELECT id FROM Lang");
  $sth->execute();
  $langs = $sth->fetchAll();
  foreach ($_POST['like-4'] as $lang) {
    $flag = true;
    foreach ($langs as $index) {
      if ($index[0] == $lang) {
        $flag = false;
        break;
      }
    }
    if ($flag == true) {
      $errors = true;
      break;
    }
  }
}
if ($_POST['check']!='on' || empty($_POST['check'])) {
  $errors = TRUE;
}

if ($errors) {
  exit();
}

try {
  $stmt = $db->prepare("INSERT INTO Person SET fio = ?, tel = ?, email = ?,
bornday = ?, gender = ?, bio = ?, checked = ?");
  $stmt->execute([$_POST['fio'], $_POST['tel'], $_POST['email'], $_POST['day'] .
'.' . $_POST['month'] . '.' . $_POST['year'], $_POST['gender'], $_POST['bio'],
true]);
  $id = $db->lastInsertId();

  $stmt = $db->prepare("INSERT INTO person_lang (id_u, id_l) VALUES
(:id_u,:id_l)");
  foreach ($_POST['lang'] as $lang) {
    $lang = $db->quote($lang);
    $stmt->bindParam(':id_u', $id_u);
    $stmt->bindParam(':id_l', $lang);
    $id_u=$id;
    $stmt->execute();
  }
}
catch(PDOException $ex){
  print('Error : ' . $ex->getMessage());
  exit();
```

```
}

header('Location: ?save=1');
```

Upload

Аналогично Include, ошибки связанные с Upload не присутствует в моём коде, в нём нет функции загрузки файлов для пользователей, поэтому уязвимости такой нет.