Отчёт о безопасности

Защита от XSS (Cross-Site Scripting)

Это тип атаки, при которой злоумышленник внедряет вредоносный скрипт в веб-страницу или приложение, которые затем выполняются на стороне клиента.

strip_tags() в PHP используется для удаления HTML и PHP тегов из строки, используется при выводе пользовательского ввода из формы. Это помогает предотвратить внедрение HTML-кода

```php
$birthDate = strtotime(strip_tags($_POST['birthDate']));
$birthDate = date('Y-m-d', $birthDate);

$name = strip_tags($_POST['name']);
$email = strip_tags($_POST['email']);
$gender = strip_tags($_POST['gender']);
$numOfLimbs = strip_tags($_POST['numOfLimbs']);
$biography = strip_tags($_POST['biography']);

setcookie('name_value', $name, time() + 30 * 24 * 60 * 60);
setcookie('email_value', $email, time() + 30 * 24 * 60 * 60);
setcookie('birthDate_value', $birthDate, time() + 30 * 24 * 60 * 60);
setcookie('gender_value', $gender, time() + 30 * 24 * 60 * 60);
setcookie('numOfLimbs_value', $numOfLimbs, time() + 30 * 24 * 60 * 60);
if (!empty($_POST['ability']))
    setcookie('ability_value', json_encode($_POST['ability']), time() + 30 * 24 * 60 * 60);
```

Защита от SQL Injection

Это вид атаки, при котором злоумышленник пытается внедрить злонамеренный SQL-код в запросы к базе данных.

Чтобы избежать SQL-инъекций, используются параметризованные запросы и подготовленных выражения ->prepare() и ->execute() для выполнения запросов к базе данных для корректной обработки параметров запроса

```php
$stmt = $db->prepare("INSERT INTO person VALUES (null,:name,:email,:birthDate,:gender,:numOfLimbs,:biography)");
$stmt->execute(['name' => $name, 'email' => $email, 'birthDate' => $birthDate, 'gender' => $gender, 'numOfLimbs' => $numOfLimbs, 'biography' => $biography]);
```

```php
$stmt = $db->prepare("INSERT INTO user VALUES (null, :pers_id , :login, :password_hash)");
$stmt->execute(['pers_id' => $pers_id, 'login' => $login, 'password_hash' => $password_hash]);
```

Защита от CSRF (Cross-Site Request Forgery)

Это вид атаки, при которой злоумышленник заставляет авторизованного пользователя совершить нежелательные действия на веб-сайте, к которому у пользователя есть доступ.

Чтобы предотвратить такие атаки, генерируется токен в сессии и проверяется его наличие и соответствие при обработке запросов. Так сервер понимает, что запрос был отправлен именно от меня

```php
$_SESSION['token'] = bin2hex(random_bytes(35));
include('form.php');
exit();
```

```php
$token = filter_input(INPUT_POST, 'token', FILTER_SANITIZE_STRING);
if (!$token || $token !== $_SESSION['token']) {
    header($_SERVER['SERVER_PROTOCOL'] . ' 405 Method Not Allowed');
    exit();
}
```

Защита от Include и Upload уязвимости

Уязвимость "Include" возникает, когда веб-приложение использует ненадежные или некорректно проверенные данные, например, пользовательский ввод, для формирования пути к файлам, которые должны быть включены в исполняемый код. Злоумышленник может внедрить злонамеренный код или получить доступ к конфиденциальным файлам на сервере.

Уязвимость "Upload" возникает, когда веб-приложение позволяет пользователям загружать файлы на сервер без должной проверки и контроля. Это может привести к загрузке вредоносных файлов, которые могут выполняться на сервере или быть доступными для скачивания другими пользователями.

Указывается путь к файлу в операторе include() вручную мы предотвратить включение нежелательных файлов

С уязвимостью "Upload" в нашей работе мы не сталкиваемся

```
require 'ability.php';
```

```
include('form.php');
```