1. Министерство науки и высшего образования Российской Федерации
2. Санкт-Петербургский политехнический университет Петра Великого
3. —
4. Институт прикладной математики и механики
5. Кафедра «Информационная безопасность компьютерных систем»

**Лабораторная работа №2**

**Разработка интернет приложения. Безопасность.**

1. по дисциплине «Разработка интернет-приложений»
2. Выполнили
3. студенты гр.  43609/3 Корольков А.А

<*подпись*>

1. Романюк Ю.В.

<*подпись*>

1. Проверил
2. преподаватель Печенкин А.И.

<*подпись*>

1. Санкт-Петербург
2. 2019

# Формулировка задания

Разработать сайт и настроить веб-сервер (безопасность). Реализованный сайт должен иметь:

1. поиск по странице;
2. авторизацию;
3. наличие ключевых особенностей, описанных в 1 лабораторной работе;
4. наличие CSS, JavaScript.

# Результаты работы

В ходе выполнения 1 лабораторной работы было принято решение разрабатывать онлайн сервис коротких текстовых заметок. В качестве отличительных особенностей выступают:

* возможность шифрования отдельных заметок на стороне пользователя;
* визуальное восприятие приближающихся deadline: чем ближе установленный срок выполнения, тем ближе ее цвет рамки к красному.

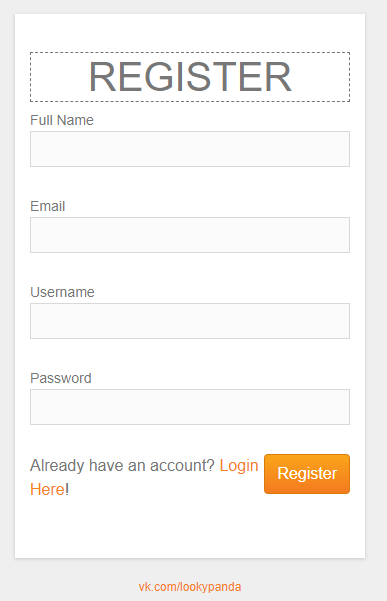


Рисунок 1- Окно регистрации

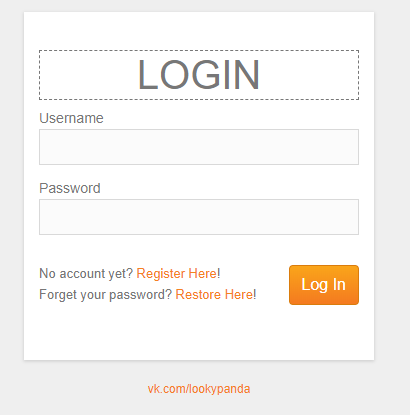


Рисунок 2- Окно авторизации

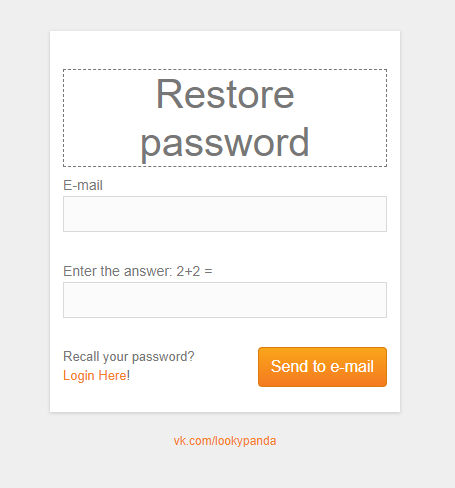


Рисунок 3 - Окно восстановления пароля



Рисунок 4 -Основное окно сайта

Алгоритм работы веб приложения:

* Пользователь выполняет регистрацию по сайте: предоставляет свой e-mail, который может быть использован для восстановления пароля; логин и пароль. Имеется проверка на повторное использование логина.
* Пользователь выполняет авторизацию: ввод пары логин-пароль. Выполняется проверка наличия в базе данных данной пары. Если данные присутствуют в базе, то выполняется авторизация пользователя и перенаправление на основную страницу сайта.
* Для добавления новой заметки пользователю необходимо ввести данные в меню, расположенном в левой части экрана. Пользователю предоставляется возможность ввести пароль и зашифровать заметку, а также ввести дату deadline.
* Заметка отправляется в базу и отображается в окне браузера.
* Для удаления заметки необходимо нажать на иконку корзины.
* Для того чтобы увидеть зашифрованную заметку, необходимо нажать на иконку замка. Для расшифрования заметки ввести пароль и нажать кнопку Decrypt.

Дополнительные опции:

* В левой части экрана расположен календарь с отмеченными на нем датами deadline. Данная информация полезна при формировании новой заметки: не перегружать день множеством заданий.
* Возможность отсортировать заметки по дате создания и по дате deadline.
* Возможность поиска по заметкам.
* Каждая заметка в зависимости от deadline подсвечивается различными цветами: чем ближе deadline, тем ближе ее цвет рамки к красному.

Настройка веб-сервера:

1. Скрываем лишнюю информацию

* По умолчанию сервер выдает слишком много информации. Например в заголовках сервера, а так же при выводе стандартных служебных страниц типа 404 можно видеть подобное Apache/2.2.12 (Win32) DAV/2 mod\_ssl/2.2.12 OpenSSL/0.9.8k mod\_autoindex\_color PHP/5.3.0 mod\_perl/2.0.4 Perl/v5.10.0.

В файле конфигурации httpd.conf команда "ServerTokens" указывает на объем показываемой информации о сервере.  
Данное значение было изменено на ServerTokens Prod.  
В данном случае выдается только название сервера.

* Заголовок X-Powered-By появляется когда информация выводится с помощью PHP скрипта. Если запросить статический html документ, то этого заголовка не будет, но если информацию на страницу выведет PHP скрипт, то в этом заголовке будет подобное X-Powered-By: PHP/5.2.5. То есть показывает информацию о том, что страница создана с помощью PHP конкретной версии.

В файле конфигурации php.ini был найден участок expose\_php = on. Значение было изменено на off.

1. Настройка phpadmin

Доступ в phpmyadmin осуществляется через пользователей базы данных, используя их логины и пароли. Сразу после установки в базе будет создан пользователь root без пароля, имеющий полный административный доступ к базе данных.

По умолчанию phpmyadmin настроен на работу именно с пользователем root без пароля, поэтому вход в phpmyadmin происходит без ввода логина и пароля с максимальными правами.

* Установлен пароль пользователю root и pma. На странице phpmyadmin во вкладке "Привилегии".
* В конфигурационном сайте config.inc.php:

Настройка авторизации и входа:

$cfg['Servers'][$i]['auth\_type'] = 'config';

$cfg['Servers'][$i]['user'] = 'root';

$cfg['Servers'][$i]['password'] = 'xxxxx';

Запрет входить под пользователями без установленных паролей.

$cfg['Servers'][$i]['AllowNoPassword'] = false;

Далее был выполнен анализ сайта с помощью анализатора Yasca. Отчет:

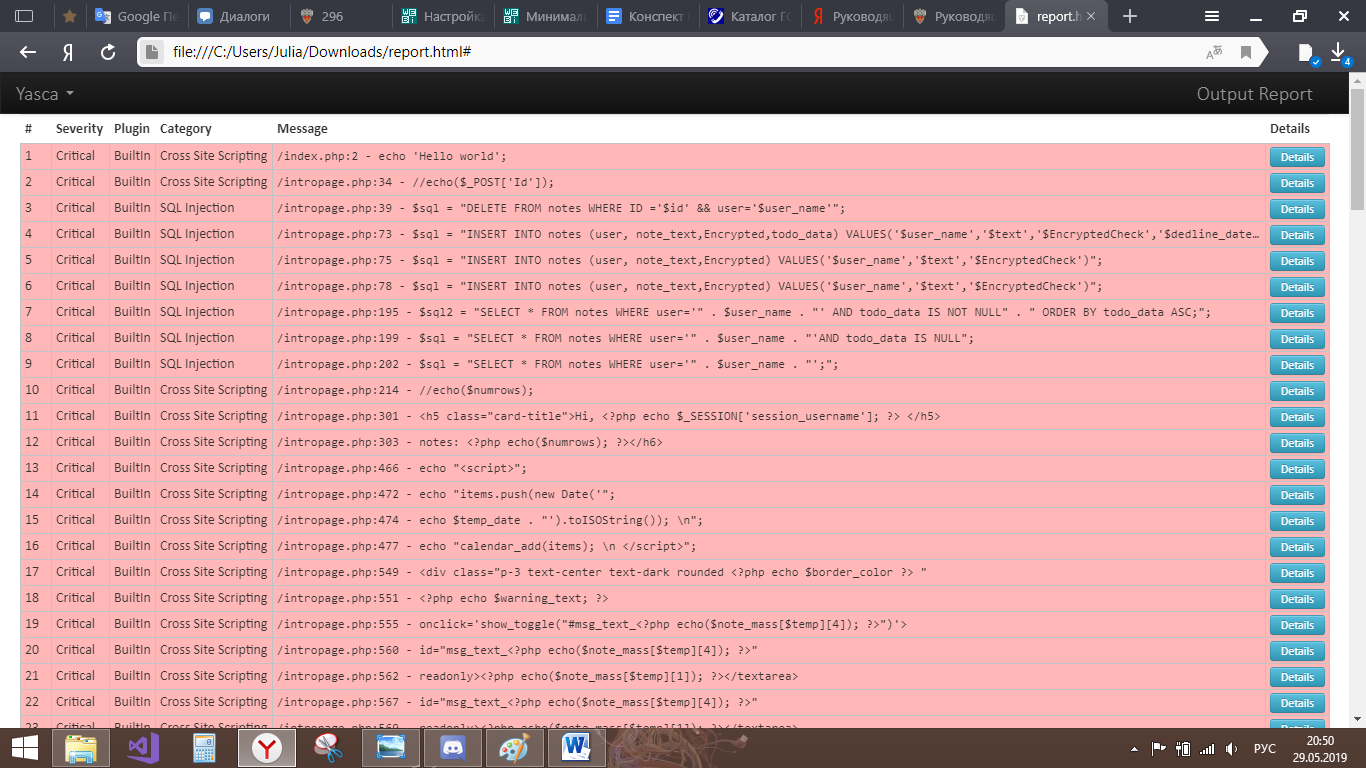


Рисунок 5 – Отчет анализатора

Проанализировав отчет можно сделать вывод, что основными уязвимостями разработанного сайта являются: SQL Injection, Cross Site Scripting, Weak Credentials. Рассмотрим каждую уязвимость и поймем как от нее можно избавиться.

SQL Injection- один из распространённых способов взлома сайтов, работающих с базами данных, основанный на внедрении в запрос произвольного SQL-кода. SQL Injection, в зависимости от типа используемой СУБД и условий внедрения, может дать возможность атакующему выполнить произвольный запрос к базе данных (например, прочитать содержимое любых таблиц, удалить, изменить или добавить данные), получить возможность чтения и/или записи локальных файлов и выполнения произвольных команд на атакуемом сервере.

Пример уязвимого кода с разработанного сайта:

$user\_name = $\_SESSION['session\_username'];  
$id = $\_POST['Id'];  
$sql = "DELETE FROM notes WHERE ID ='$id' && user='$user\_name'";  
$query = mysqli\_query($con, $sql) **or** trigger\_error(mysqli\_error($con) . " in " . $sql);

В данном случае изменяем данный участок кода на:

$input\_text = trim($\_SESSION['session\_username']);  
$input\_text = htmlspecialchars($input\_text);  
$input\_text = mysqli\_real\_escape\_string ($con,$input\_text);  
$user\_name = $input\_text;  
  
$input\_text2 = trim($\_POST['Id']);  
$input\_text2 = htmlspecialchars($input\_text2);  
$input\_text2 = mysqli\_real\_escape\_string ($con,$input\_text2);  
$id = $input\_text2;  
$sql = "DELETE FROM notes WHERE ID ='$id' && user='$user\_name'";  
$query = mysqli\_query($con, $sql) **or** trigger\_error(mysqli\_error($con) . " in " . $sql);

Для избавления от этой уязвимости были использованы встроенные PHP функции.

 В HTML некоторые символы имеют специальное значение и для сохранения своего значения должны быть преобразованы в HTML сущности. Функция htmlspecialchars возвращает строку, над которой проведены некоторые из таких преобразований. Эта функция полезна при отображении данных, введенных пользователем, которые могут содержать нежелательные HTML тэги.

Mysqli\_real\_escape\_string - экранирует специальные символы в строке для использования в инструкции SQL с учетом текущей кодировки соединения

Weak Credentials. Данный класс атак может быть реализован большим числом способов, но цель у них одна – получение доступа к учётной записи пользователя системы или администратора.

Для реализации защиты от атак данного типа в зависимости от конкретного случая возможно применение самых разных методов защиты: многофакторная аутентификация, проверка надёжности паролей, установление периодичности смены паролей, использование ограничения на количество попыток входа пользователя в систему и таймаута между неудачными попытками, использование менеджеров сессий для генерации идентификаторов и т.д.

Пример уязвимого кода из разработанного сайта:

$password=$\_POST['password'];

В данном случае изменяем данный участок кода на:

$input\_text4 = trim($\_POST['password']);  
 $input\_text4 = htmlspecialchars($input\_text4);  
 $input\_text4 = mysqli\_real\_escape\_string ($con,$input\_text4);  
 $password=$input\_text4;  
 // Validate password strength  
 $uppercase = preg\_match('@[A-Z]@', $password);  
 $lowercase = preg\_match('@[a-z]@', $password);  
 $number = preg\_match('@[0-9]@', $password);  
 $specialChars = preg\_match('@[^\w]@', $password);  
  
 **if**(!$uppercase || !$lowercase || !$number || !$specialChars || strlen($password) < 8) {  
 $message='Password should be at least 8 characters in length and should include at least one upper case letter, one number, and one special character.';  
 }**else**{  
 //echo 'Strong password.';

Cross Site Scripting- одна из разновидностей атак на веб-системы, которая подразумевает внедрение вредоносного кода на определенную страницу сайта и взаимодействие этого кода с удаленным сервером злоумышленников при открытии страницы пользователем.

Пример уязвимого кода с разработанного сайта:

<h5 class="card-title">Hi, **<?php   
 echo** $\_SESSION['session\_username']; **?>** </h5>

В данном случае изменяем данный участок кода на:

<h5 class="card-title">Hi, **<?php** $input\_text = trim($\_SESSION['session\_username']);  
 $input\_text = htmlspecialchars($input\_text);  
 $input\_text = mysqli\_real\_escape\_string($con, $input\_text);  
 **echo** $input\_text; **?>**</h5>

Так же был установлен флаг HttpOnly. Этот Флаг делает клиентские куки недоступными через языки сценариев, такие как JavaScript. Данная настройка активируется— в php.ini : session.cookie\_httponly = True

Так же был проанализирован топ 10 уязвимостей OWASP.

* 1. Инъекции. В разработанном приложении были найдены и исправлены уязвимости SQL инъекций и инъекции кода. Примеры исправления были приведены в отчете выше.
  2. Недостатки аутентификации. Была реализована проверка надёжности создаваемых паролей: ограничение на длину в 8 символов, требования наличия цифр, заглавных и прописных букв и специальных символов. Выполняется аннулирование идентификаторов сессий при отсутствии активности пользователя или его выходе из системы или браузера.
  3. Разглашение конфиденциальных данных. Для предотвращения атак типа «человек посередине» сайт был переведен на принудительное использование https. Так же в качестве особенности разработанного приложения реализовано шифрование заметок на стороне клиента.
  4. Внешние сущности XML (XXE). Разработанный сайт не подвержен данной атаке в виду отсутствия функционала работы с файлами и XML документами.
  5. Недостатки контроля доступа. Для исправления уязвимостей манипуляций с метаданными (повторное воспроизведение или подмена cookie-файлов) был включен флаг делающий куки недоступными через языки сценариев, такие как JavaScript. Так же при попытке зайти на страницы сайта без авторизации происходит перенаправление на страницу с ней.
  6. Некорректная настройка параметров безопасности. Для этого была скрыта лишняя информация сервере и его ПО в выдаваемых страницах, был настроен доступ в phpmyadmin с надежным паролем и запрещен доступ для пользователей без пароля.
  7. Межсайтовое выполнение сценариев (XSS**)**. Защита от данного типа атак приведена выше.
  8. Небезопасная десериализация.В разработанном приложении отсутствует работа с файлами или сериализация информации .
  9. Использование компонентов с известными уязвимостями. Наличие данной уязвимости характерно для приложений, использующих старые версии фреймворков, зависимостей или компонентов. В нашем приложении используются актуальные версии фреймворков и прочего ПО.
  10. Недостатки журналирования и мониторинга.Для мониторинга и журналирования используются встроенные в веб-сервер и сервер базы данных возможности. Так же встроена поддержка средства яндекс метрики для сбора статистики.

# выводы

В результате выполнения лабораторной работы был разработан онлайн сервис коротких текстовых заметок. В качестве отличительных особенностей реализованы возможность шифрования отдельных заметок на стороне пользователя; и визуальное восприятие приближающихся deadline: чем ближе установленный срок выполнения, тем ближе ее цвет рамки к красному.

При написании сайта мы столкнулись с множеством сложностей, которые связаны с плохим знанием языков: php, javascript, html и CSS .